DOCUMENT RESUME

ED 031 274 LI 001 640

By-King, Paul Allen, Jr.
A Novel Solid State Character Generator.
Massachusetts Inst. of Tech., Cambridge. Electronic Systems Lab.
Spons Agency-Council on Library Resources, Inc., Washington, D.C.
Report No-ESL-TM-386
Pub Date Jun 69
Note-109p.
EDRS Price MF-$0.50 HC-$5.55
Descriptors-*Display Systems, Information Systems, *Input Output Devices, *Man Machine Systems, *Media Technology, *Orthographic Symbols
Identifiers-Project Intrex, *Sinusoidal Scan

This thesis describes a character generator which was built for a refreshed cathode-ray tube (CRT) display and demonstrates the feasibility of displaying high quality dot-matrix characters using the low-band width sinusoidal scan. The CRT used was a low-cost entertainment-quality television tube, and hence had a very poor deflection band width. Settling time for the CRT system was about 10 microseconds. A technique was developed and made operational which displayed a whole dot-matrix character in a time which was less than one settling time of the display system. It is an application of the existing technique called sinusoidal scan to the display of dot-matrix characters. A data compression technique was also invented and implemented. It allows up to a 75 percent reduction in the size of the stored character tables which are normally associated with dot-matrix displays. The technique involves a digital algorithm which fills in dots between given dots in a matrix. This technique proved quite effective in producing good quality characters, and is economically attractive. (Author/RM)

TECHNICAL MEMORANDUM Copy No.______

ESL-TM-386

A NOVEL SOLID STATE CHARACTER GENERATOR

by

Paul Allen King, Jr.

June, 1969

Research Grant from
Council on Library Resources

This Technical Memorandum consists of the unaltered thesis of Paul Allen King, Jr. submitted in partial fulfillment of the requirements for the degree of Master of Science at the Massachusetts Institute of Technology in June, 1969. The research reported herein was made possible through the support extended the Massachusetts Institute of Technology, Project Intrex, at the Electronic Systems Laboratory under a research grant from the Council on Library Resources, Inc. This grant is designated as M.I.T. DSR Project Number 27808.

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

Electronic Systems Laboratory
Department of Electrical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

A NOVEL SOLID STATE CHARACTER GENERATOR

by

PAUL ALLEN KING, JR.

S.B., Massachusetts Institute of Technology
(1969)

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June, 1969

Signature of Author Allen King
Department of Electrical Engineering, May 23, 1969

Certified by James K. Roberge
Thesis Supervisor

Accepted by ______________________________
Chairman, Departmental Committee oh Graduate Students

A NOVEL SOLID STATE CHARACTER GENERATOR

by

PAUL ALLEN KING JR.

Submitted to the Department of Electrical Engineering on May 23, 1969 in partial fulfillment of the requirements for the Degree of Master of Science.

ABSTRACT

This thesis describes a character generator which was built for a refreshed cathode-ray tube display. The CRT used was a low-cost entertainment-quality television picture tube, and hence had a very poor deflection bandwidth. Settling time for the CRT system was about 10μs.

A technique was developed and made operational which displayed a whole dot-matrix character in a time which was less than one settling time of the display system. It is an application of the existing technique called sinusoidal scan to the display of dot-matrix characters.

A data compression technique was also invented and implemented. It allows up to a 75 percent reduction in the size of the stored character tables which are normally associated with dot-matrix displays. The technique involves a digital algorithm which fills in dots between given dots in a matrix. This technique proved quite effective in producing good quality characters, and is economically attractive.

THESIS SUPERVISOR: James K. Roberge
TITLE: Assistant Professor of Electrical Engineering

ACKNOWLEDGMENT

The research reported herein was made possible through the support extended the Massachusetts Institute of Technology, Project INTREX, at the Electronic Systems Laboratory under a research grant from the Council on Library Resources, Inc. and designated as M.I.T. DSR Project Number 27808.

The author wishes to express his sincere thanks to Professor James K. Roberge for his guidance during the course of this work. Thanks are also due to Alfred DiPillo, who built the circuits involved, to Mrs. Arlene Zamagni, who typed the manuscript, and to Norman Darling who drew the figures.

TABLE OF CONTENTS

page

CHAPTER I. THE PROBLEM 9

1.1 Intrex System 9

1.2 Character Generator for the Console 11

1.3 Advantage of Sinusoidal Scan 12

1.4 Scanning Character Generators 13

1.5 Weaknesses of Scanning Generators 19

CHAPTER II. DISPLAYING DOT-MATRIX CHARACTERS USING SINUSOIDAL SCAN 26

2.1 Segmenting a Sinusoid 26

2.2 Sinusoid Distortions 27

2.3 Horizontal Double Frequency 27

2.4 Correcting Vertical Distortion 30

CHAPTER III. CHARACTER SMOOTHING 35

3.1 Size of Matrix 35

3.2 Smoothing Algorithm 39

3.3 Interpolation Algorithm 40

3.4 Simple Interpolation 41

3.5 Strong Interpolation 45

CHAPTER IV. HARDWARE IMPLEMENTATION 48

4.1 Data Section 48

4.1.1 Simple "Formattor" 48

4.1.2 Column Interpolation 50

4.1.3 Simple Interpolation 52

4.1.4 Order Reversing 59

4.1.5 Strong Interpolation 60

4.2 Timing Chain 71

page

CHAPTER IV. HARDWARE IMPLEMENTATION (Con't.)

4.2.1 Horizontal Timing 73

4.2.2 Phase-Locked Loop 74

4.2.3 Vertical Deflection Circuits 78

4.2.4 Vertical Timing 83

4.3 Read-Only Memory 85

4.3.1 Memory of the Scanning Generators 85

4.3.2 Storing Skeletal Matrices 86

4.3.3 Modified ASCII Code 87

4.3.4 Address Map 88

4.3.5 Key Word 89

CHAPTER V. CONCLUSION 94

5.1 Comparison of Dot-Matrix and Scanning Generators 94

5.2 Evaluation of the Interpolation Algorithms 99

TABLE OF FIGURES

page

CHAPTER I. THE PROBLEM

1.1 Sinusoidal Scan 14

1.2 Monoscope 16

1.3 Flying Spot Scanner 17

1.4 Complete System 18

1.5 Jitter Mechanism 20

1.6 Omission Mechanism 20

1.7 Fidelity of Scanned Characters 22

1.8 Mask Designed for a Specific Scan Pattern 24

CHAPTER II. DISPLAYING DOT-MATRIX CHARACTERS USING SINUSOIDAL SCAN

2.1 Order of Displaying Dots 28

2.2 Matrix Displayed Using Sinusoidal Scan 29

2.3 Family of Scan Patterns 31

2.4 Derivation of Maximum Frequency 33

2.5 Matrix Displayed with Corrected Scan 34

CHAPTER III. CHARACTER SMOOTHING

3.1 Typical Dot-Matrix Characters 36

3.2 Most Probable Patterns in a 9 by 7 Matrix 38

3.3 Type of Display Points 41

3.4 Simple Interpolation 44

3.5 Moving the Vertex of an Oblique Angle 46

3.6 Examples of Interpolation 47

CHAPTER IV. HARDWARE INTERPOLATION **page**

4.1 Dot-Matrix Generator Block Diagram 49

4.2 Digitally Controlled Switch 51

4.3 Shift-Register Converter 51

4.4 Effect of Rule No. 1 and No. 2 53

4.5 Interpolator for Rule No. 1 and No. 2 53

4.6 Interpolator for Rules No. 1 - No. 4 54

4.7a,b Effect of Rules No. 1 - No. 4 56

4.7c Contents of Shift Registers 57

4.7d Patterns in the Window Matrix for Simple Interpolator 58

4.8 Bit-Reversing Shift Registers 61

4.9 Shift-Register Network for Rules No. 1 through No. 5 63

4.10 Data Present in Interpolator 64

4.11 Undefined Values in Window Matrix 64

4.12 Window Matrices Definition 66

4.13 Logic Diagram of Shift-Register Network 67

4.14 Patterns in the Window Matrix for Strong Interpolation 69

4.15 Interpolation Logic Diagram 70

4.16 Blanking Amplifier 72

4.17 Phase-Locked Loop 75

4.18 System Response of Phase-Locked Loop 77

4.19 Open Loop Frequency Response 79

4.20 Horizontal Generator 80

4.21 Tuned Amplifiers 81

4.22 SHIFT Oscillator 84

4.23 Contents of ROM 90

4.24 ROM-Interpolator Interface 92

CHAPTER V. CONCLUSION page

5.1 Intrex Console 95

5.2 Dot Matrix Character Generator: Simple and Strong Interpolation 96

5.3 Monoscope with Interlace 97

5.4 Monoscope without Interlace 98

5.5 Dot Matrix Character Generator: Strong Interpolator 100

5.6 Dot Matrix Character Generator: Simple Interpolator 101

5.7 Dot Matrix Character Generator: No Interpolator 102

5.8 Sample Text Using Dot-Matrix 105

5.9 Sample Text Using Monoscope 106

CHAPTER I

THE PROBLEM

1.1 INTREX SYSTEM

The primary objective of Project Intrex is to evolve a functional design for an automated library system that could become operational at M.I.T. and elsewhere during the 70's. The core program encompasses four major areas of activity:

1. The Augmented Catalog - The development of a computer-assisted bibliographic search system.
2. Guaranteed Full Text Access - The development of a system to make the full text of every document available at remote stations.
3. Network Integration - The investigation of methods of exchanging information between libraries.
4. Fact Retrieval - The development of a system for retrieving data from very large files.

Of these, only the Augmented Catalog will be considered here. There are three areas of activity with the Augmented Catalog program:

1. Augmented-Catalog Inputting
2. Storage and Retrieval
3. The Display Console

In the Augmented Catalog Inputting phase, a large base of literature cataloged according to author, title, subject, etc. is being developed and translated into digital form. In the Storage and Retrieval phase, catalog input data is being transferred into disk-file storage. Programming techniques are being developed for rapid computer searches, through the literature base, for articles that will be of interest to the catalog user.[1] The Display Console is a special purpose on-line computer terminal that allows the catalog user to communicate with the processor and to direct the computer searches in an interactive mode. The Display Console consists of a keyboard, with which the user sends information to the computer, and a display device, by which the computer conveys information to the user. It is the Display Console, in a sense, that makes the Augmented Catalog useful. If the user had to communicate with the computer by means of punched cards and computer printout, there would be little point in utilizing a computer search: he could probably find the necessary information more quickly by the conventional methods in a library.

In operation, the user types in appropriate descriptor words, subject heading, or author names. The computer then searches the literature base for articles categorized under these headings and indicates to the user the number of articles found. If this number is too large, the user can reduce it by typing in additional constraints. When the search is complete, the user may elect to have titles or abstracts of the articles displayed at the console.

The display element at the console is a large screen cathode-ray tube (CRT). This type of output medium is desirable because it eliminates the long and noisy delays associated with mechanical typewriter terminals. Considerations of system requirements led to the existing display system which has the following specifications:

1) The display capacity is approximately 1700 characters -- there are 31 lines of 56 characters per line.
2) The character set includes both upper and lower case English letters (the standard 96 character ASCII code).

3) The character set also includes an additional 96 symbols including Greek letters and mathematical symbols.

4) The character set is alterable, so that characters or character styles can be changed.

5) The Display Console must be economically reproducible.

6) The display system has the provision for selective erasure, to facilitate editing and making corrections.

This last feature imposes a heavy burden on the console electronics. Since storage tubes cannot be selectively erased, a conventional cathode-ray tube had to be employed. Information must be constantly refreshed on the face of the tube, one frame after another, in a manner similar to a television set. There are problems associated with regenerating the 1700 character display rapidly enough to prevent noticeable flicker.

1.2 CHARACTER GENERATOR FOR THE CONSOLE

The character generator is one of the most important parts of the console. The characters are the only thing the user sees: If the characters are not clear, all the elegant features of the system are lost because they cannot be read easily. Two previous attempts at generating readable characters have resulted in only mediocre characters. This thesis involves a third character generator which was designed to give much better performance. To understand the design, it is first necessary to understand the Intrex display system. The following paragraphs explain how the current configuration was chosen,[2] and what effects that structure has on the possible design of a character generator.

The console system consists of a small computer and a magnetic drum which services up to ten indivdual display consoles. The local computer services the demands of the ten consoles. It is capable of reading a console keyboard, reading and writing on the magnetic drum, and communicating with the time-sharing computer. All information displayed on the CRT of each console is stored on the magnetic drum. Thus the cost of refresh storage and centralized data processing equipment is amortized over ten consoles. The data on the drum passes through the character generator located at the console and is displayed on a screen in the following manner: A format generator produces a raster of over 1700 character positions every 17 milliseconds. The generator is synchronized to the magnetic drum so that one complete revolution of the drum corresponds to one complete raster on the CRT. Consequently, drum addresses correspond to specific positions on the CRT screen, resulting in a savings of hardware.[3] Note that this scheme requires the character generator illuminate a whole character on the CRT screen every time the drum sends one (every 8.5μs). The character generator is constantly a slave to the codes coming from the drum.

1.3 ADVANTAGE OF SINUSOIDAL SCAN

Much of the expense of a large capacity refreshed alphanumeric display is taken up by the CRT and its associated deflection circuits.[1] To reduce this cost, an inexpensive television picture tube is being used. The tube is magnetically deflected, with a deflection-system settling time of about 10μs. This means that the beam cannot be moved quickly to an arbitrary point on the screen since time has to be allowed for the beam to settle. The current solution is to selectively intensify a fixed scan pattern. The pattern, which was chosen because it is immune to the settling time of the beam, is generated by adding a sine wave to the vertical deflection signal, and applying a ramp to the horizontal deflection.[4] Both patterns can be implemented easily with hardware because they both have very narrow bandwidth. Figure 1.1 shows the pattern with the correct intensification to produce the word "Intrex".

The vertical scan pattern is immune to settling time because in the sinusoidal-steady-state analysis, a time delay is merely a phase lag. This lag, if it is constant, can be corrected in theory by an appropriate phase advance of the signal to the deflection circuit, although in practice, it is effected by a phase delay in an alternate path. The ramp, which is applied to the horizontal deflection, sweeps a full line of 56 characters without stopping. Since ample time (about 20μs) is allowed at the beginning of each line for the beam to stabilize its transient response, limited horizontal deflection bandwidth is tolerable.

It is quite compelling to stay with this sinusoidal scan pattern for its simplicity and ease of implementation. It is also the only effective way in which one can display characters with deflection circuits that require 10μs to settle. Thus vector generators (which have variable scan patterns) and the type of dot matrix generators which rely on digital spot positioning are truly not feasible.

Constrained to using the sinusoidal scan pattern, there are two major ways to generate the necessary blanking signal. One way involves electrically scanning an image of the character set which has been physically constructed in the device. The other method, described here, stores the intensity patterns in a digital read-only memory. The patterns for both are, of course, displayed on the sinusoidal scan.

1.4 SCANNING CHARACTER GENERATORS

There are three types of scanning generators, the flying spot scanner,

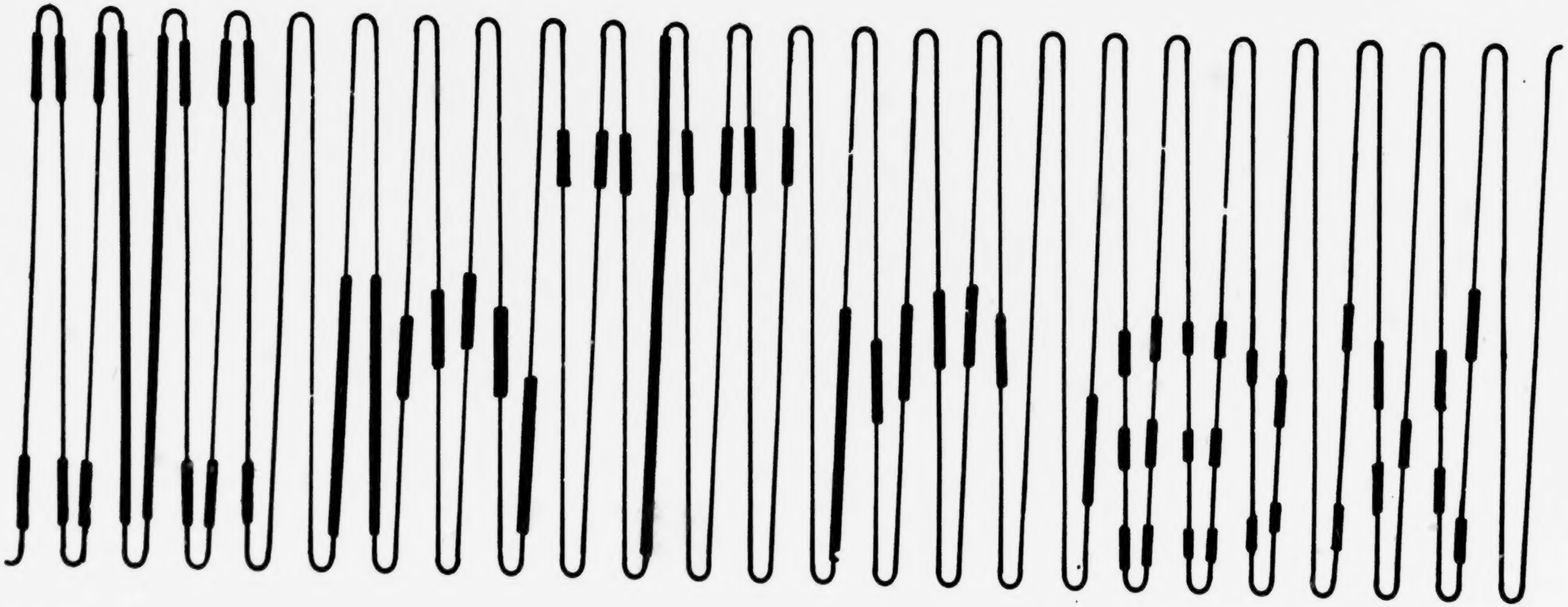


Fig. 1.1 Sinusoidal Scan

the monoscope, and the vidicon tube. Much work has been done at Intrex by Paul McKenzie to develop the flying-spot scanner,[5] and by Professor James K. Roberge and the author to develop the monoscope. The vidicon tube was not developed because of its short operational lifetime. The following paragraphs explain each type of scanning generator and present the causes for their partial failure.

The monoscope generates the intensification pattern from ink characters on a metal plate.[6] An electron beam is electrostatically deflected to the desired spot in the desired character. If the beam strikes ink, the secondary emission of the plate is less. A ring collects the secondary electrons--the current they produce is amplified and used as the video signal. The process is shown diagrammatically in Figure 1.2.

A flying-spot scanner in its simplest form, consists of a small electrostatically deflected CRT, a Photo Multiplier Tube (PMT), and a character mask between them. (See Figure 1.3.) The beam is directed to the desired position of the appropriate character, and a spot of light is produced on the phosphor, directly behind the character. If the position of the character is clear, light is picked up by the photo multiplier tube. The PMT signal is amplified and is used as the blanking signal for the display tube.

The video signal from either generator is used to form characters on the display CRT in the following manner. The character code from the drum directs the beam in the scanning generator to the desired character. At the same time, the beam of the display CRT is positioned where the character is to be illuminated. (See Figure 1.4.) The sinusoidal scan pattern is formed concurrently on both the scanning generator and the

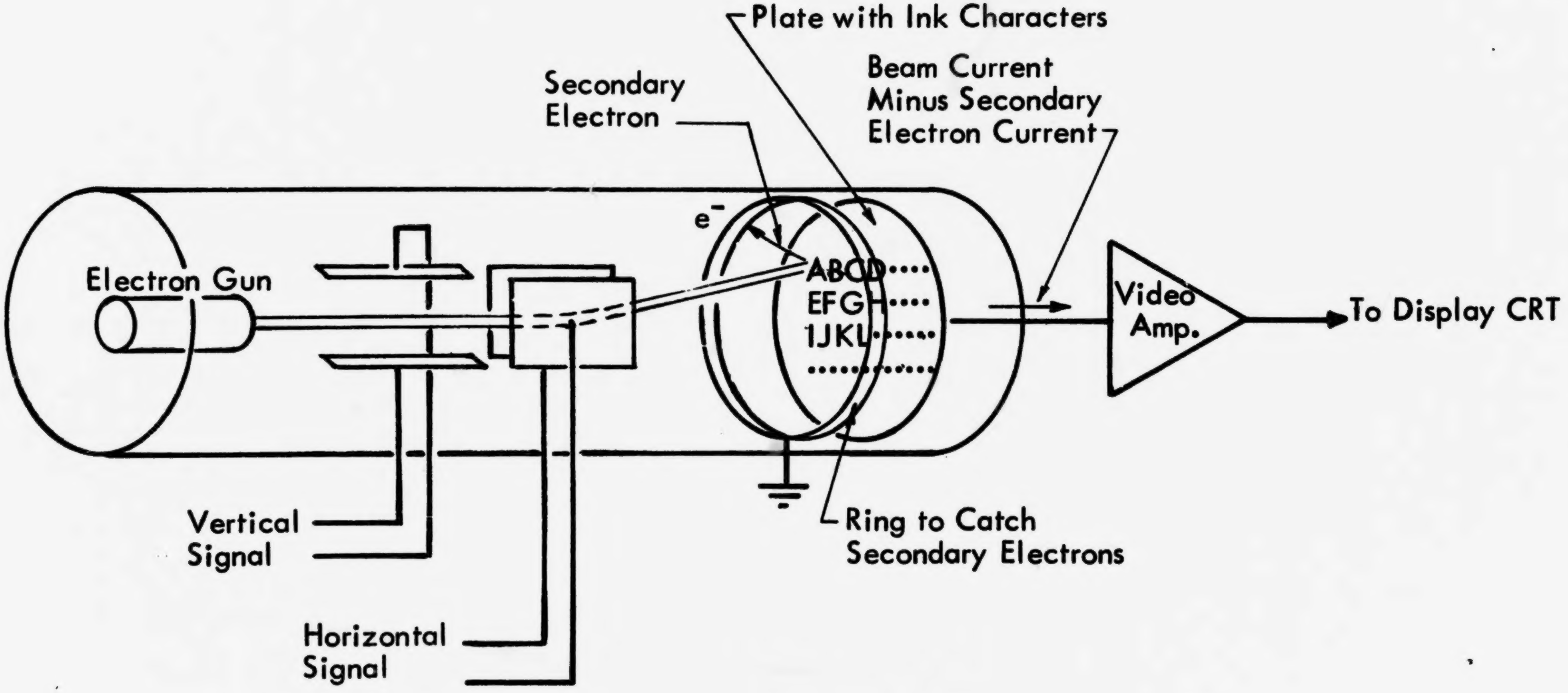


Plate with Ink Characters
Secondary Electron
Beam Current Minus Secondary Electron Current
e^-
Electron Gun
ABCD
EFGH
IJKL
Video Amp.
To Display CRT
Vertical Signal
Horizontal Signal
Ring to Catch Secondary Electrons

Fig. 1.2 Monoscope

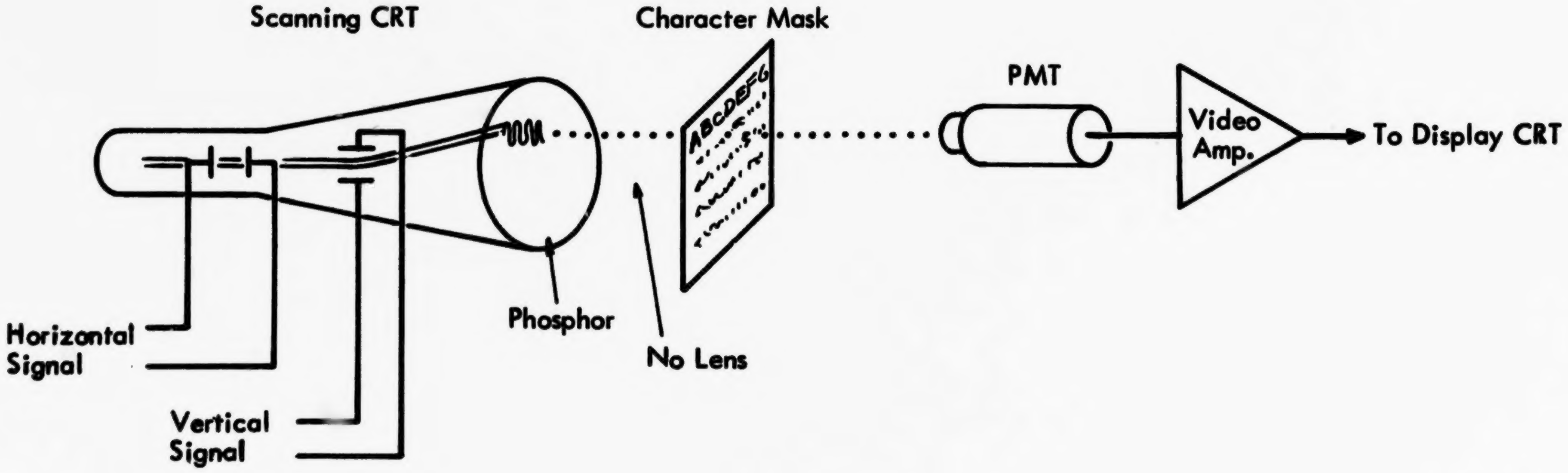

Scanning CRT
Character Mask
PMT
Video Amp.
To Display CRT
ABCDEFG
Phosphor
No Lens
Horizontal Signal
Vertical Signal

Fig. 1.3 Flying Spot Scanner

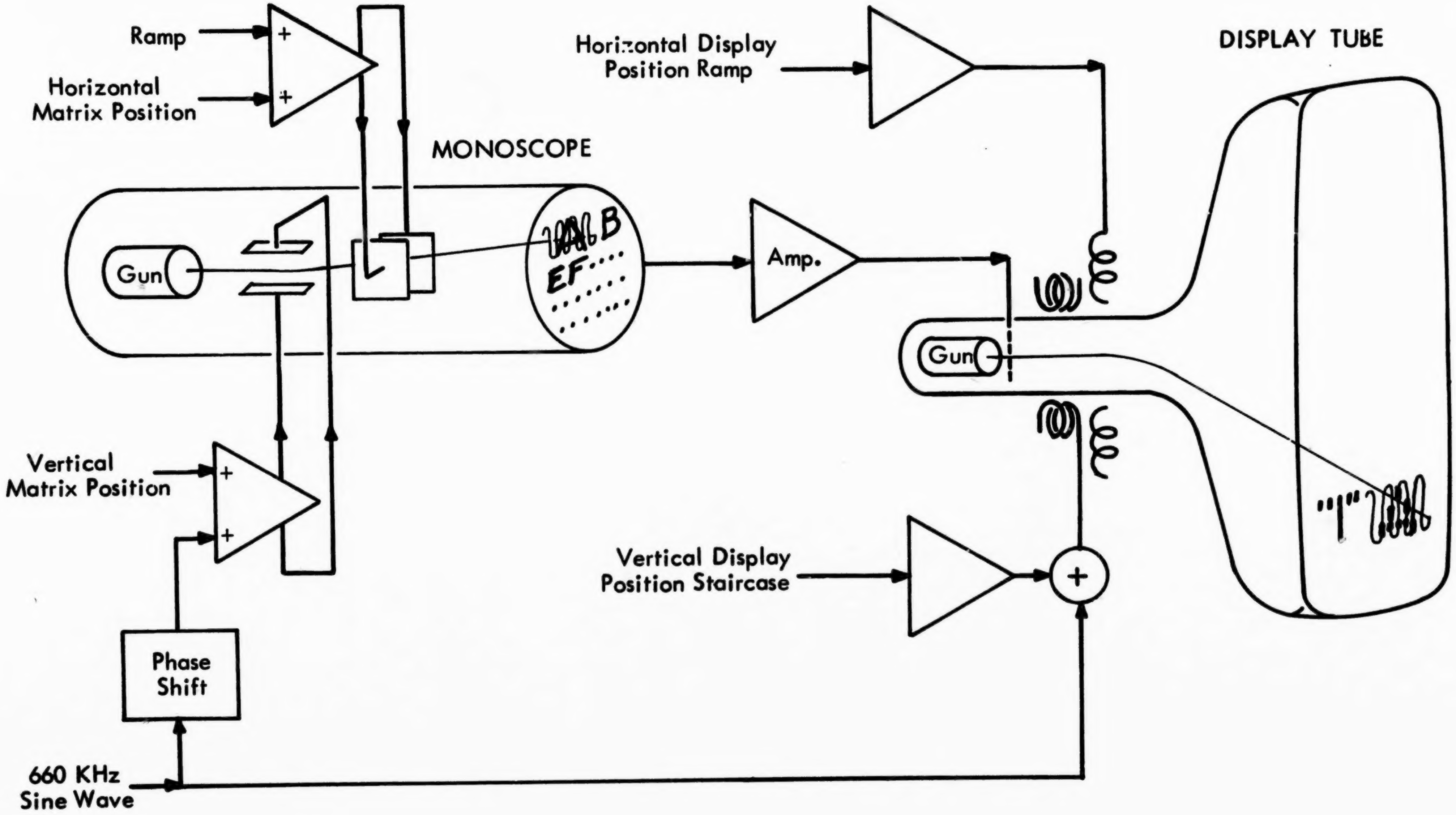

Ramp
Horizontal Matrix Position
+
+
MONOSCOPE
Gun
Vertical Matrix Position
+
+
Phase Shift
660 KHz Sine Wave
Horizontal Display Position Ramp
Amp.
Vertical Display Position Staircase
+
Gun
DISPLAY TUBE

Fig. 1.4 Complete System

display CRT. Throughout the scan, the coordinates of both beams are the same, relative to the starting point of the scan. The video signal which is produced from the scanning generator is used as the blanking signal for the display. Consider the monoscope system for example. Wherever the monoscope plate has an ink spot, there is less secondary emission and less current in the collector ring. This signal is amplified to produce a voltage which intensifies the beam at the spot corresponding to the position of the ink on the monoscope character. Thus every dark area (on the monoscope plate) which is traversed by the beam produces a corresponding bright area on the display screen. Characters are transferred by the scan from a position on the mask to a position on the screen.

1.5 WEAKNESSES OF SCANNING GENERATORS

The characters generated by the scanning generator tend to flicker. If the beam passes too close to the edge of the character, it may or may not intensify during that scan. (See Figure 1.5.) Magnetic fields bend the electron beam of the character generator and change the relative position of the character. Thus a scan line which falls just on the edge of a character might be pushed off the edge and not intensify the display. Any power supply ripple or noise which finds its way into the deflection input or the anode of the generator tube will have the same shifting effect. The shifting will cause one element of the character to flicker on and off. The effect can be quite annoying since some of the pickup is due to the 60 cycle power line frequency. Because the refresh rate is 57 cycles, the noise will beat with the refresh rate to produce a 3 cycle per second flicker.

Noise in the video amplifier may boost a marginal signal over the threshold. The monoscope in particular has a low level output (1mV) and

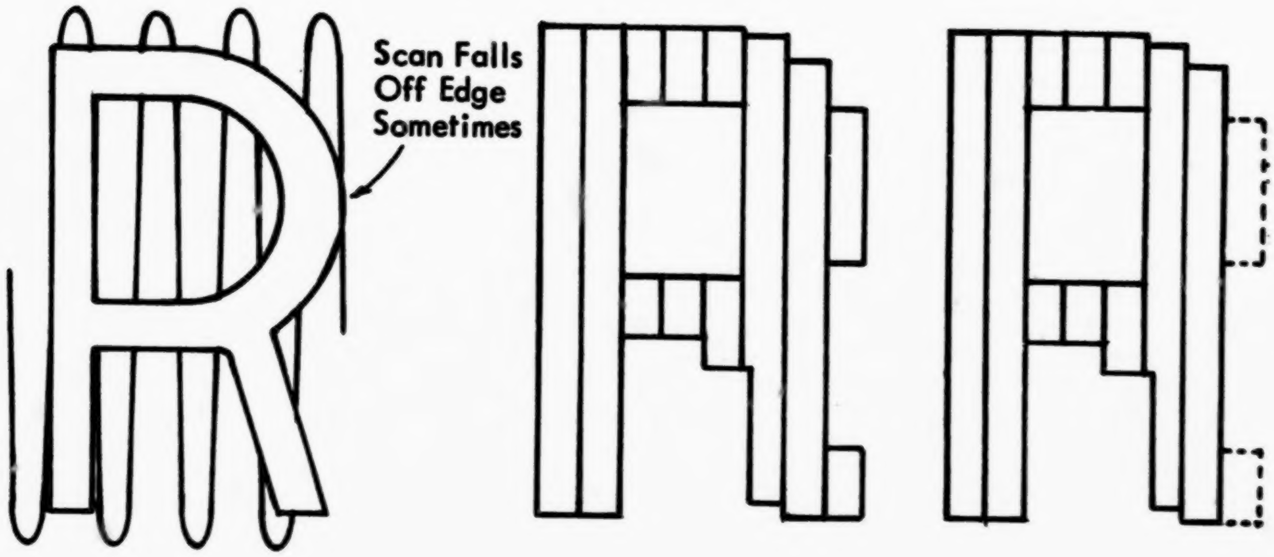

Scan Falls
Off Edge
Sometimes

Fig. 1.5 Jitter Mechanism

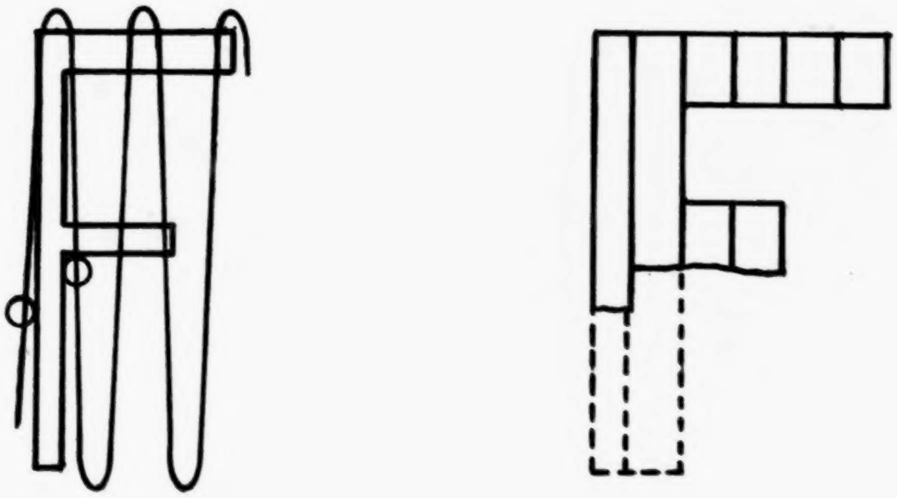


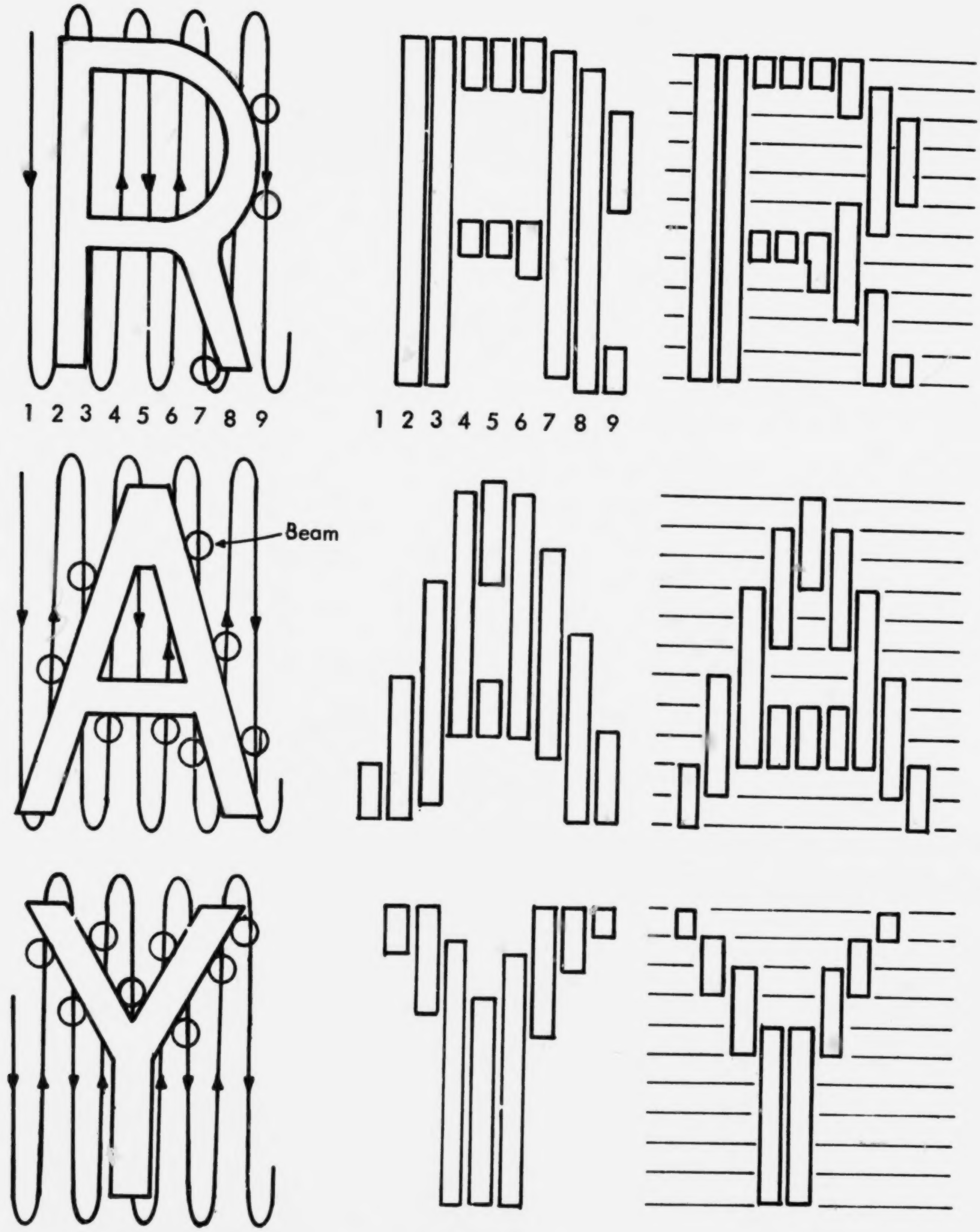
Fig. 1.6 Omission Mechanism

thus noise can be a problem. Since some of the noise is correlated with the 60 cycle line frequency, 3 cycle beating occurs.

There seems to be another inherent weakness in scanning generators. The character set for any scanning generator must be constructed so that good characters will result no matter what scan pattern is used, or how it is positioned over the character. It is possible to have a coarse scan pattern in which two consecutive scan lines straddle some vertical feature of a character (e.g. the vertical bar of the letter "F" as in Figure 1.6). The result is that the vertical feature would not be displayed. This symptom is noticed when the monoscope is used to generate characters on the display console.

The usual adjustment which one makes when the above omissions occur is to brighten the scanning beam. This has the effect of making the scanning spot larger (as well as increasing the signal to noise ratio). The larger spot causes less precise scanning information. Many little details of the letter, such as corners and curves, are filled in by the larger beam. Figure 1.7 shows the letters "R", "A", and "Y" as they appear on the character mask with the sinusoidal pattern superimposed. Circles indicate the beam diameter. This diameter is chosen so that an intensify signal is sent to the display if the beam is closer than this diameter to an edge. Figure 1.7b shows the display produced by the scan of Figure 1.7a. Figure 1.7c shows an intensify pattern which was chosen to produce the most readable characters possible. It seems apparent that operated in this manner, the scanning generators do not always generate the optimum blanking signal.

The persistence of the phosphor on the flying-spot scanner degrades


1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
Beam

(a) Mask (b) Scanned (c) Optimal

Fig. 1.7 Fidelity of Scanned Characters

the image in a similar manner. The moving spot has a tail of glowing phosphor which makes the spot appear to be shaped like a comet.

One might try to custom tailor the character set to the particular scan pattern used. Figure 1.8 shows the mask characters with choppy edges to compensate for the distortion caused by finite beam size. The problem with this idea is that there is no mechanism for positioning of the beam over the character set. Beam position is a function of anode and accelerating voltages, magnetic fields in the tube, and amplifier gains and offsets. In addition, since the basic timing for the Intrex display is derived from the magnetic drum which varies speed slightly, the phase and amplitude of the sine wave used for the sinusoidal scan varies. Thus the pattern is not constant.

The display pattern is actually alternated between two sine waves 180° out of phase. The idea is similar to interlacing on a television, and is necessary to achieve enough resolution to produce readable characters. Yet if the two scans produce different intensities for the same general area, an annoying 30 cycle flicker is produced.

Digitally stored images do not have these disadvantages because all processing is done digitally. One problem which does exist is that transitions are allowed to occur only at specified places in a sweep of the sine wave. This limitation is not present in the scanning generators, which can make an intensity transition at any place in a scan. Yet both methods, the image scanning and the digitally stored, do limit the horizontal resolution to the number of lines scanned. Thus the limited vertical resolution of the dot pattern might not be as critical as supposed. Note that the "optimum" characters of Figure 1.7 were actually generated

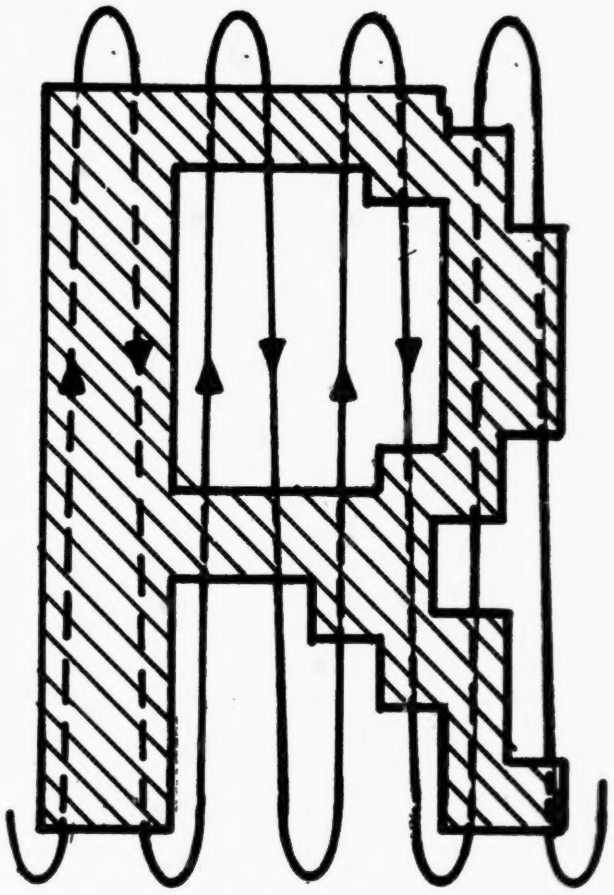


Fig. 1.8 Mask Designed for a Specific Scan Pattern

from a 9 by 11 digital matrix. In addition, the digital generator is fast enough to display more sine waves per character. Thus the horizontal resolution is actually better than the scanning generator.

CHAPTER II

DISPLAYING DOT-MATRIX CHARACTERS USING SINUSOIDAL SCAN

The digitally-stored character generator offers tremendous possibilities for Project Intrex. MOS read-only memories are becoming available, and should be economically feasible before any large-scale production of the console is undertaken. As argued in the previous chapter, and demonstrated in Chapter 5, digitally stored images produce better characters than scanning generators do.

The design of the character generator was strongly biased by the necessity to display the matrix using the sinusoidal scan pattern compatible with the Intrex console. In this chapter, the features of the sinusoidal scan relevant to the design will be discussed.

One essential feature of this type of scanning is that it is a fixed scan pattern which does not change from character to character. The only way in which two characters can be made to look different on the screen is to intensify the beam in different portions of the pattern.

2.1 SEGMENTING A SINUSOID

The following observation allows one to plot a matrix using the sinusoidal scan: A sine wave can be crudely divided into display sections and adjoined curved sections. Where the display region is defined to end and the curved region begins is a subjective question, and depends on the arguments to be developed. The display region includes all zero-crossings where the second derivative of the sine wave is zero, and thus it is reasonably straight. This linear display region can be segmented by the character generator into discrete equal intervals of time. Each interval corresponds to one dot of the matrix, and is intensified if that dot is on. Note that line segments, not points, are displayed on the CRT. The beam does not stop moving to display a dot.

One column of a matrix can be displayed in each display region of the sine wave. An $\underline{m}$ column matrix can be displayed in m/2 cycles of the sine wave since one cycle has 2 linear regions. Figure 2.1 shows a 13 by 9 display matrix with some of its points numbered in the order in which they will be displayed using the sinusoidal scan.

2.2 SINUSOID DISTORTIONS

There are two characteristics of a sine wave which tend to distort the squareness of a matrix displayed in the above manner. First, adjacent display regions do not have the same slope, and thus they are closer together in either the top or the bottom of the scan. Therefore the horizontal distance between two dots varies in the displayed matrix. Secondly, the sine wave moves more slowly near the top and bottom in its curved sections. The dots tend to get bunched up near the top and bottom of the scan, and spread out in the center, causing the vertical distance between two points to vary. The magnitude of these aberrations are a function of how much of the sine wave is used to plot the matrix. Some 7 by 5 matrices are shown in Figure 2.2 as they would appear on the CRT screen for various values of $\underline{p}$, the fraction time used for display. The squares represent the actual position used for display, and the dots are arranged in a rectangular grid for reference.

2.3 HORIZONTAL DOUBLE FREQUENCY

The first of the distortions can be reduced significantly. The solution is to change the scan pattern slightly by superimposing another sine wave onto the horizontal ramp signal. If the horizontal sine wave is twice

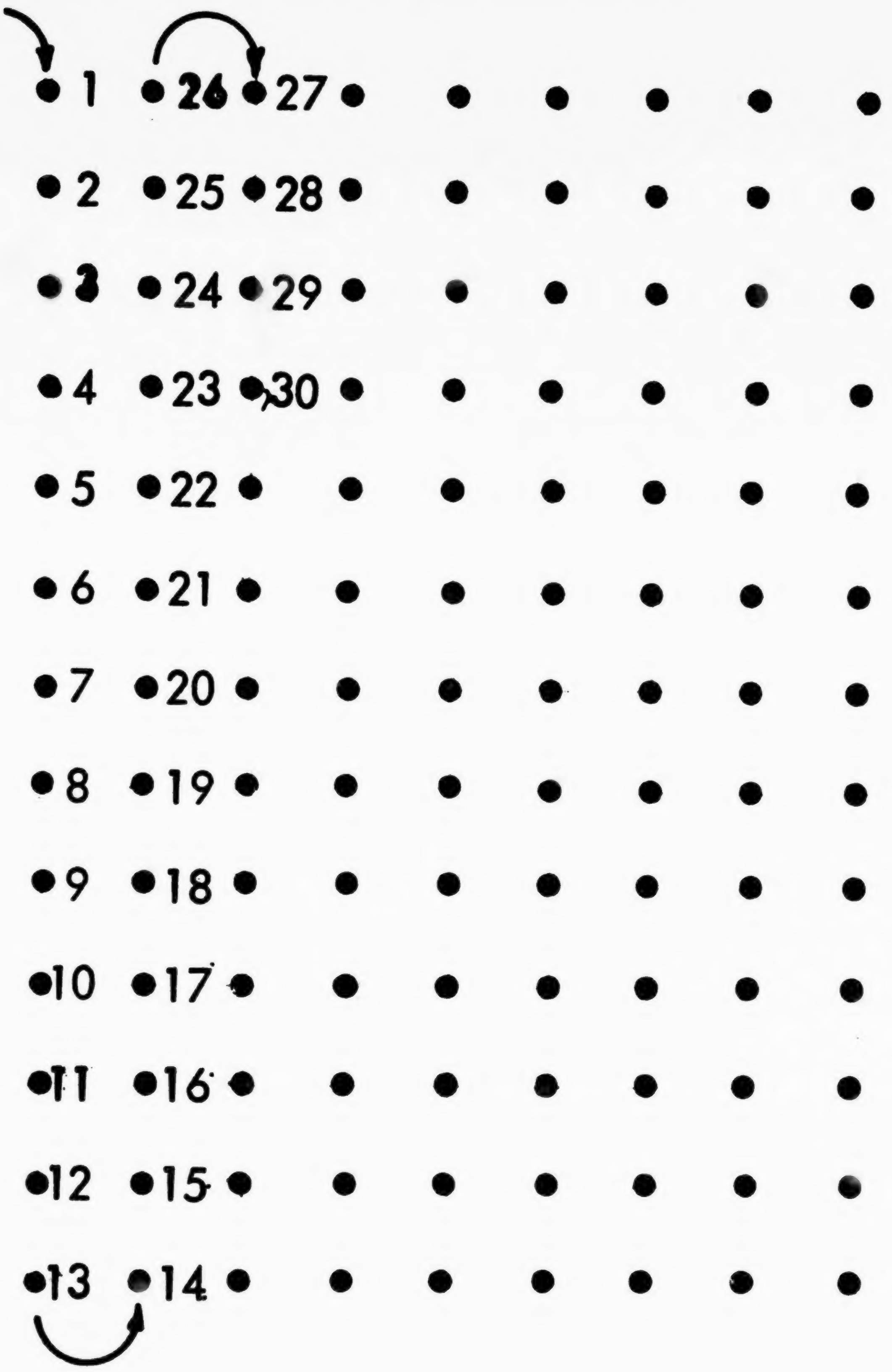
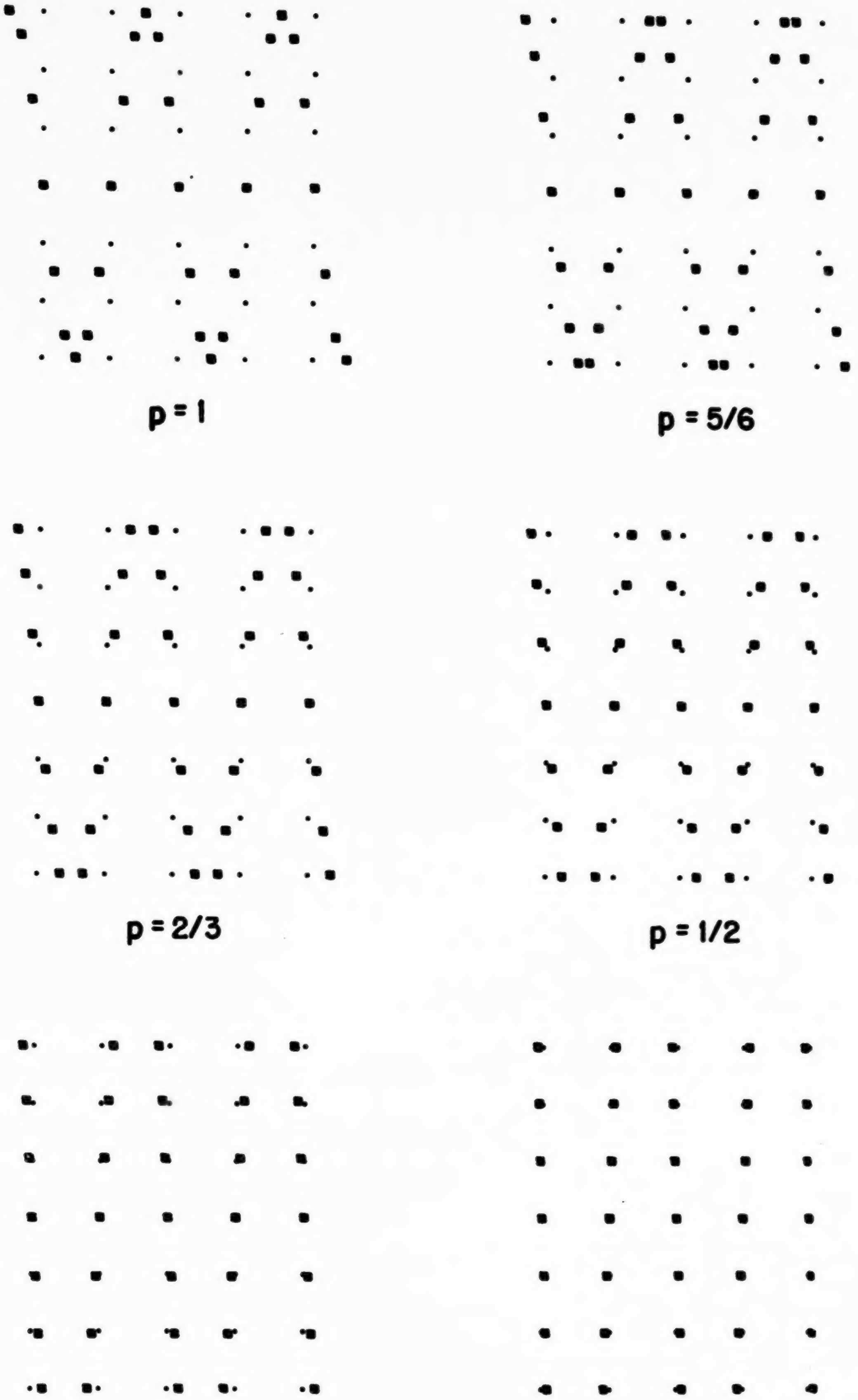

1
26
27
2
25
28
3
24
29
4
23
30
5
22
6
21
7
20
8
19
9
18
10
17
11
16
12
15
13
14

Fig. 2.1 Order of Displaying Dots



$p = 1$ $p = 5/6$

$p = 2/3$ $p = 1/2$

$p = 1/3$ $p = 1/6$

Fig. 2.2 Matrix Displayed Using Sinusoidal Scan

the frequency of the vertical sine wave, and it has the correct phase and amplitude, the scan pattern produced will have display regions which are almost vertical. Figure 2.3 shows the parametric curves generated by:

$$x = \sin(\omega t)$$

$$y = t + a\cdot\sin(2\omega t + b)$$

A family of curves is plotted for various values of a and b. The waves in each column have the same values of a, the amplitude, and the waves in each row have the same value of b, the phase. One very desirable scan along with many poor ones can be produced.

It should be noted in passing that these patterns have little to do with a two-termed Fourier expansion of a square wave. That expansion requires a sine wave of three times the fundamental to be added to the vertical signal. The undesirable result of this addition is that the beam has twice the velocity in its "linear" regions. This doubles the required operating frequencies of the generator.

2.4 CORRECTING VERTICAL DISTORTION

The vertical distortion can also be reduced. The most obvious solution is to use only the center portion of the sine wave where the beam velocity is relatively constant. The major problem is that it forces the operating frequency of the character generator to be higher. If one plots an m by n matrix in 8.5μs using the fraction p of the sine wave to display, then the frequency f_1 at which the dots must be displayed is given by:

$$f_1 = \frac{m\cdot n}{p\cdot(8.5\mu s)}$$

Frequency is inversely proportional to the fraction of time used to scan, p.

Consider displaying a 13 by 9 display matrix. If p is assumed to be 2/3, f is 20MHz. Since the matrix with p = 2/3 does have substantial vert-

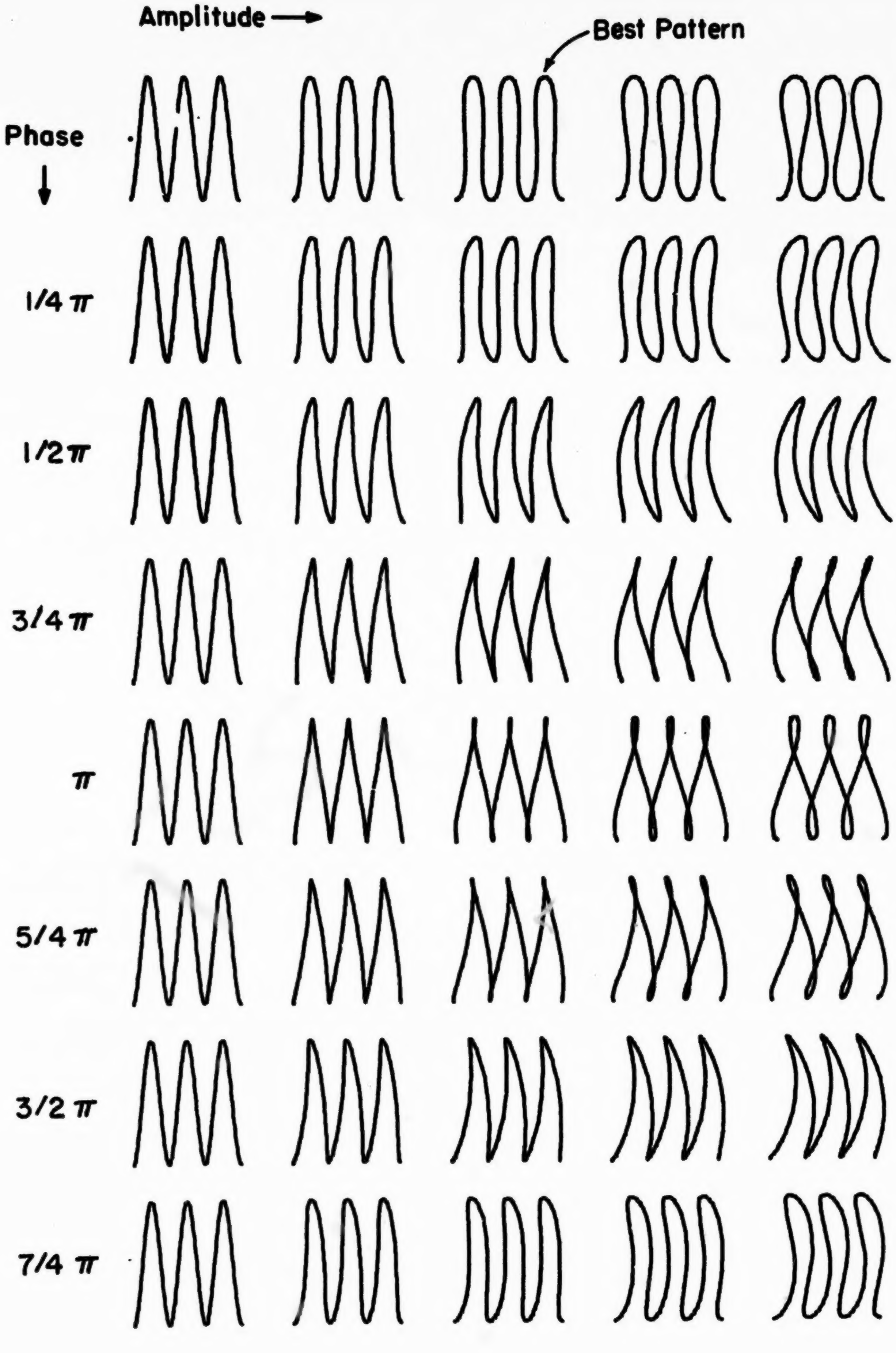

Amplitude ⟶
Best Pattern
Phase ↓
1/4 π
1/2 π
3/4 π
π
5/4 π
3/2 π
7/4 π

Fig. 2.3 Family of Scan Patterns

ical aberrations, consider reducing p to 1/3. This doubles the display frequency to 40MHz, out of the range of ordinary logic.

A better solution to the vertical distortion problem is to divide each linear region into segments of equal length on the CRT screen. Thus the time to plot each dot is not constant throughout a scan. Figure 2.4 is used in the derivation of the maximum frequency at which dots are displayed. Assume the matrix is displayed using a fraction p of the time in each cycle. Then the height of the displayed region is $2\cdot A\cdot\sin(\pi p/2)$. Assume for the moment that the scan was not a sine wave, but a ramp whose velocity was the maximum vertical velocity of the sine wave. This maximum is d/dt of $A\cdot\sin(\pi t/2T)$ evaluated at $t = n\pi$ and is $\pi A/2T$ (distance per time). The rate necessary to display m dots on a ramp of length $2\cdot A\cdot\sin(\pi p/2)$ is given by:

$$f_2 = \frac{\pi\cdot m}{4\cdot T\cdot\sin(\pi p/2)} = \frac{\pi\cdot m\cdot n}{2\cdot\sin(\pi p/2)\cdot(8.5\mu s)}$$

This is also the maximum rate of display of dots for the sinusoidal scan, because the number of dots per vertical distance is the same, and the maximum slope is the same. As would be expected, f_1 approaches f_2 when p becomes small and thus the vertical distortion vanishes.

In this section, a scan pattern has been developed which requires very simple deflection circuitry, and is capable of displaying reasonably square dot matrices quickly.

Figure 2.5 shows the same set of 7 by 5 matrices as does Figure 2.2, except the vertical and horizontal corrections have been applied. For values of p up to 2/3, the distortion is negligible, and does not affect the quality of the characters generated.

The following section will explain the circuitry which intensifies the portions of the scan pattern necessary to form a character.

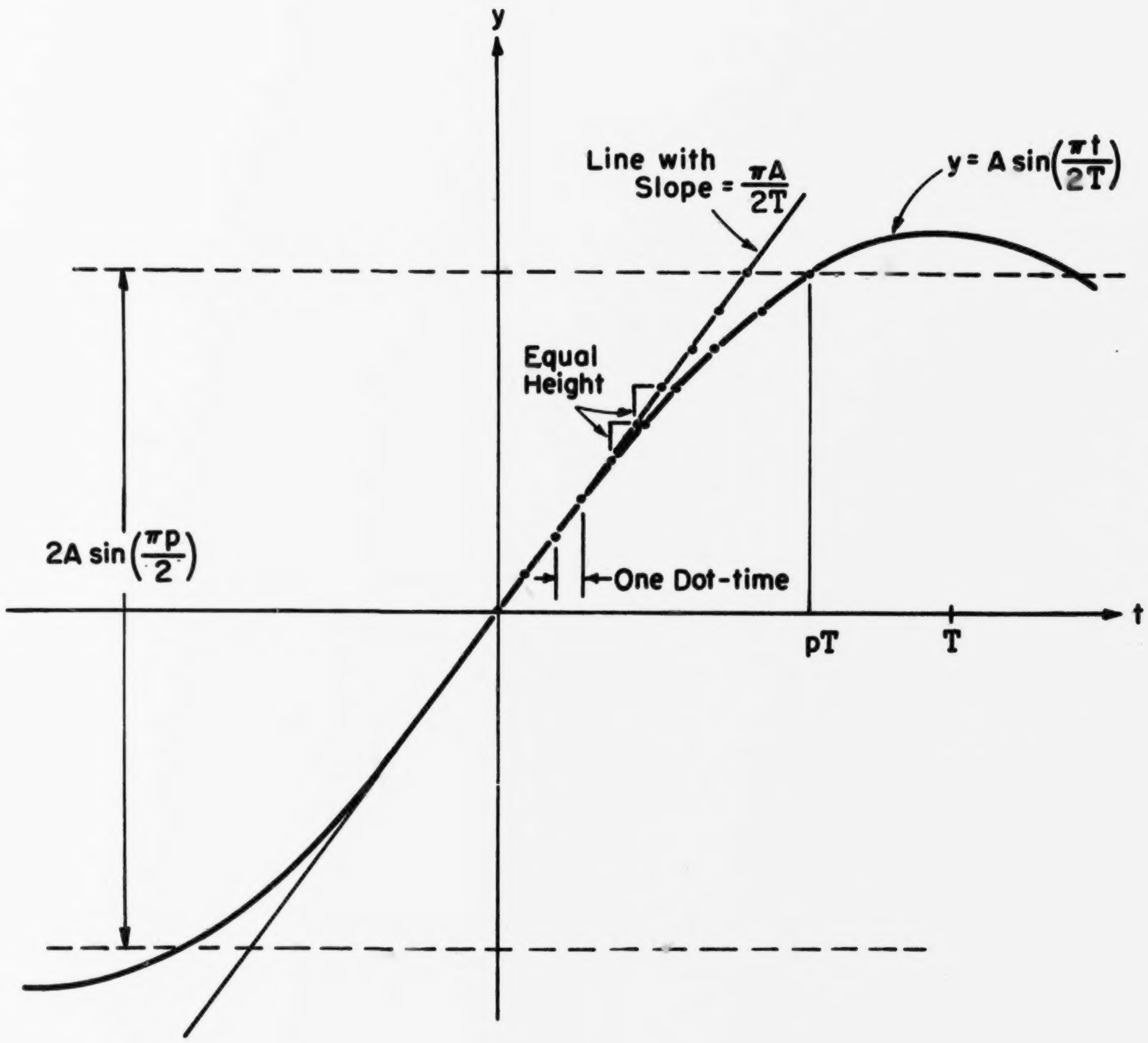
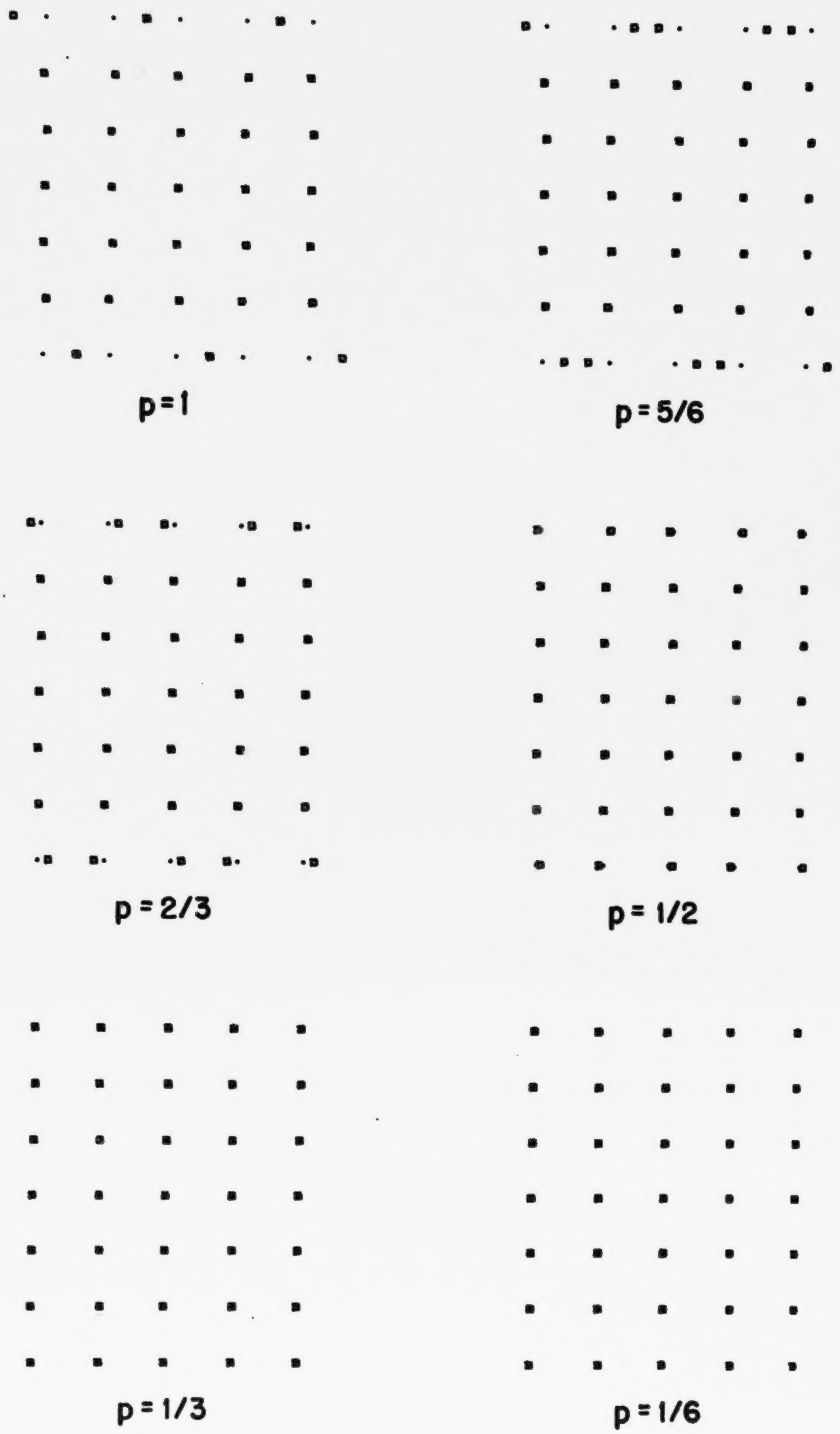

y
Line with Slope = $\frac{\pi A}{2T}$
$y = A \sin\left(\frac{\pi t}{2T}\right)$
Equal Height
$2A \sin\left(\frac{\pi p}{2}\right)$
One Dot-time
t
pT
T

Fig. 2.4 Derivation of Operating Frequency



p=1

p=5/6

p=2/3

p=1/2

p=1/3

p=1/6

Fig. 2.5 Matrix Displayed with Corrected Scan

CHAPTER III

CHARACTER SMOOTHING

3.1 SIZE OF MATRIX

The quality of the digitally stored image is strongly dependent on the number of points displayed per character. It is assumed that the information about the shape of each character is stored in a dot matrix. As shown in Figure 3.1, a 7 by 5 dot matrix* gives fair quality capital letters, while a 9 by 7 matrix gives good upper and lower case letters. Because complicated Greek letters are desired, and because the users of the console will not be used to dot-like characters, better than the 9 by 7 must be generated.

A 9 by 7 matrix has roughly twice as many dots to be stored per character as does a 7 by 5. Increasing the size even further would increase the cost of the digital memory used to store the tables. It would be a costly brute-force solution to go to a larger matrix.

If characters are plotted in a 9 by 7 matrix, there are 2^{63}, or approximately 10^{19} different dot patterns which can be displayed. This is in striking contrast with the fact that there are at most 10^3 characters which can be recognized if displayed in that format. Make the rational assumption that there are less than 10^5 ways to display a given character on the matrix in readable form. It then follows that less than one dot pattern in 10^{11} can actually be recognized as a character. 99.999999999 percent of all patterns which can appear in a 7 by 9 matrix are meaningless.

---

*Throughout this thesis, the size of a vertical column (or position in a column) is noted first, followed by the size of a horizontal row (or position in a row). This convention is adopted to conform with matrix notation. Unfortunately, it does not conform to the usual notation of placing the smaller number first, followed by the larger number.

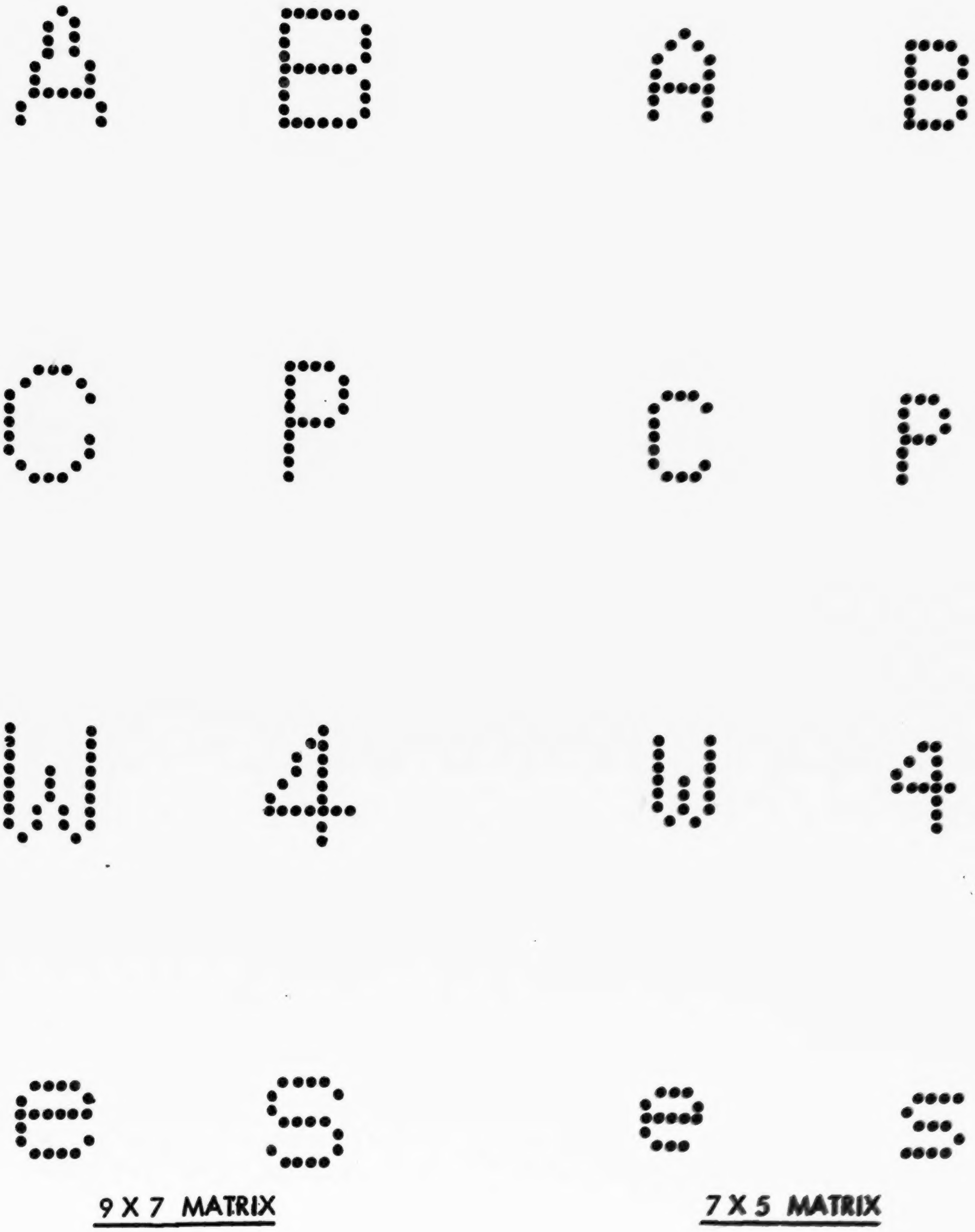

9 X 7 MATRIX
7 X 5 MATRIX

Fig. 3.1 Typical Dot-Matrix Characters

What characteristic separates the 10^8 meaningful patterns from the 10^{19} meaningless ones? Consider how characters are chosen: They have evolved over the ages under the necessity that they be readable. If the human eye is not constructed to easily recognize a character, that character is probably meaningless. The eye is probably good at recognizing lines and curves, but is poor at recognizing sets of dots randomly sprinkled into a matrix. A large fraction of the 10^{19} meaningless patterns were found to resemble dots randomly scattered into a matrix. The rest of the matrices resemble backward characters, or conglomerations of character parts. Figure 3.2 shows some matrices which were generated by computer using a random number generator. They represent a random sampling of the 10^{19} patterns which could be constructed. None of them even closely resemble any character.

If one could weed out all the random patterns, and never allow them to be stored in the memory, the memory could be made smaller. In particular, it might be possible to store the m-bit matrices for all readable characters in less than m bits of memory, say n bits. This means that most of the m bit matrices, cannot be stored in the memory. If the n bits are coded in a way not yet defined, most of the non-representable codes will be the random patterns. More importantly, there will be at least one n-bit code for each readable character.

All of the above is obviously possible. The ASCII code is one such 7-bit code which can specify one of 128 "m-bit" matrices. The problem is that there is no simple way to map a given ASCII character into the appropriate m-bit matrix. If the map plus the storage of the n-bit codes involves more logic than storing the m-bit matrix directly, clearly nothing is gained.

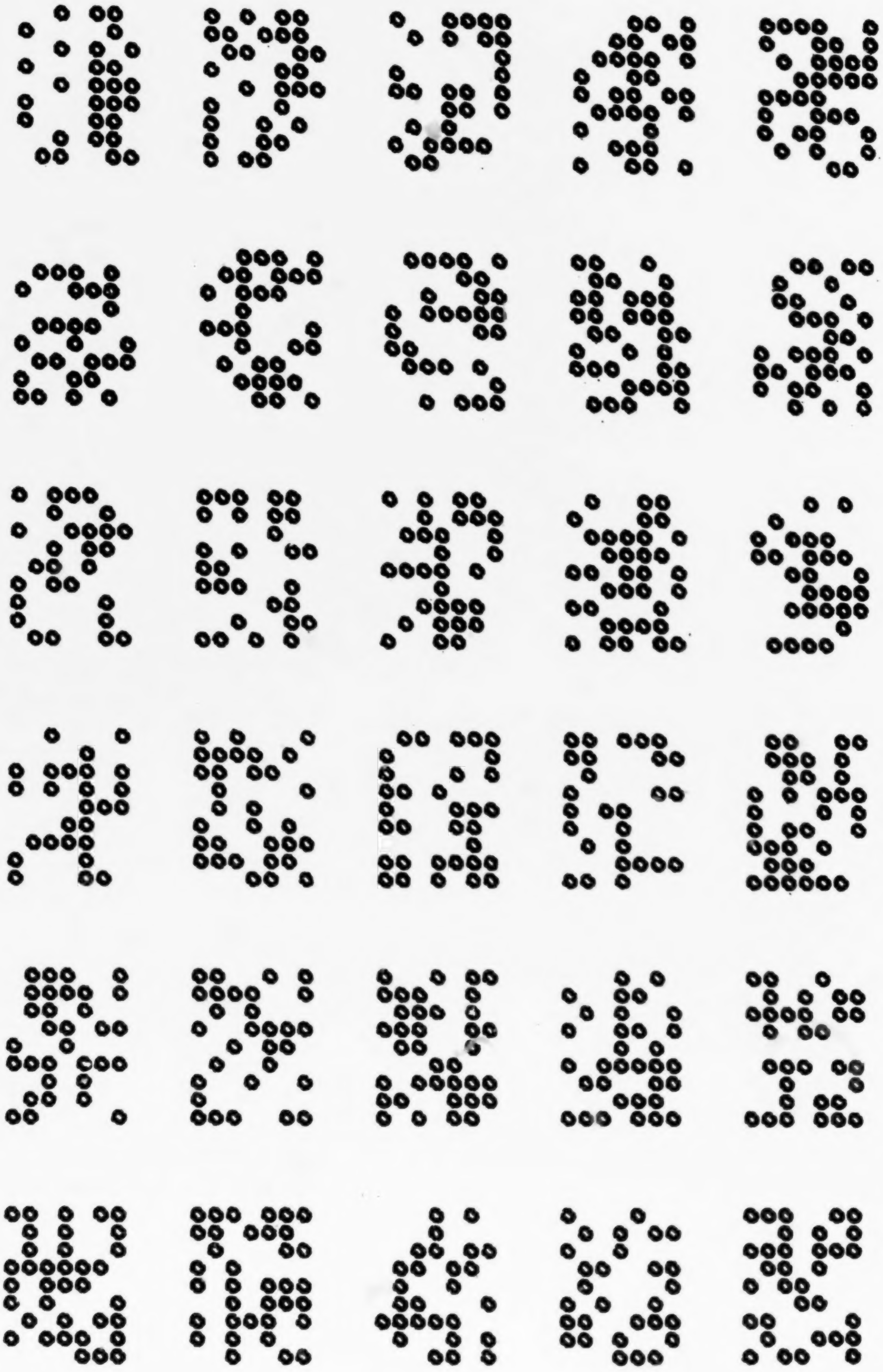


Fig. 3.2 Most Probable Patterns in a 9 X 7 Matrix

The task is to find some simple mapping which will save memory. Many schemes were investigated. Line and curve drawing techniques tend, in the final implementation, to require many bits to specify endpoint, slope and curvature information. In addition, they are not compatible with the Intrex fixed sinusoidal scan pattern.

If one looks at one column at a time of the display matrix, an interesting fact is evident: Certain combinations of bits just never occur. It might therefore be possible to code the ones which can occur in fewer bits, using a column by column application of the principle described above for matrices. The technique is promising for capital letters, but gets involved for lower case and Greek letters. No simple map could be found to reduce the number of permissible columns.

3.2 SMOOTHING ALGORITHM

One would surmise that smoothing should best be done on a "two-dimensional" basis, since the characters exist on flat surfaces. A simple two-dimensional algorithm becomes evident if one compares characters generated on different size matrices. (See Figure 3.1.) The advantage which one gains when a larger matrix is used seems to be that the corners of the characters are more rounded and oblique lines are less ragged. Being able to finely position the parts of a letter (e.g. the middle bar of an "E") does not seem to be very critical. The eye is accustomed to recognizing both handwriting, which has inherently variable dimensions, and printing, which is done in many different fonts. Little information is conveyed by the exact placement of a character's parts.

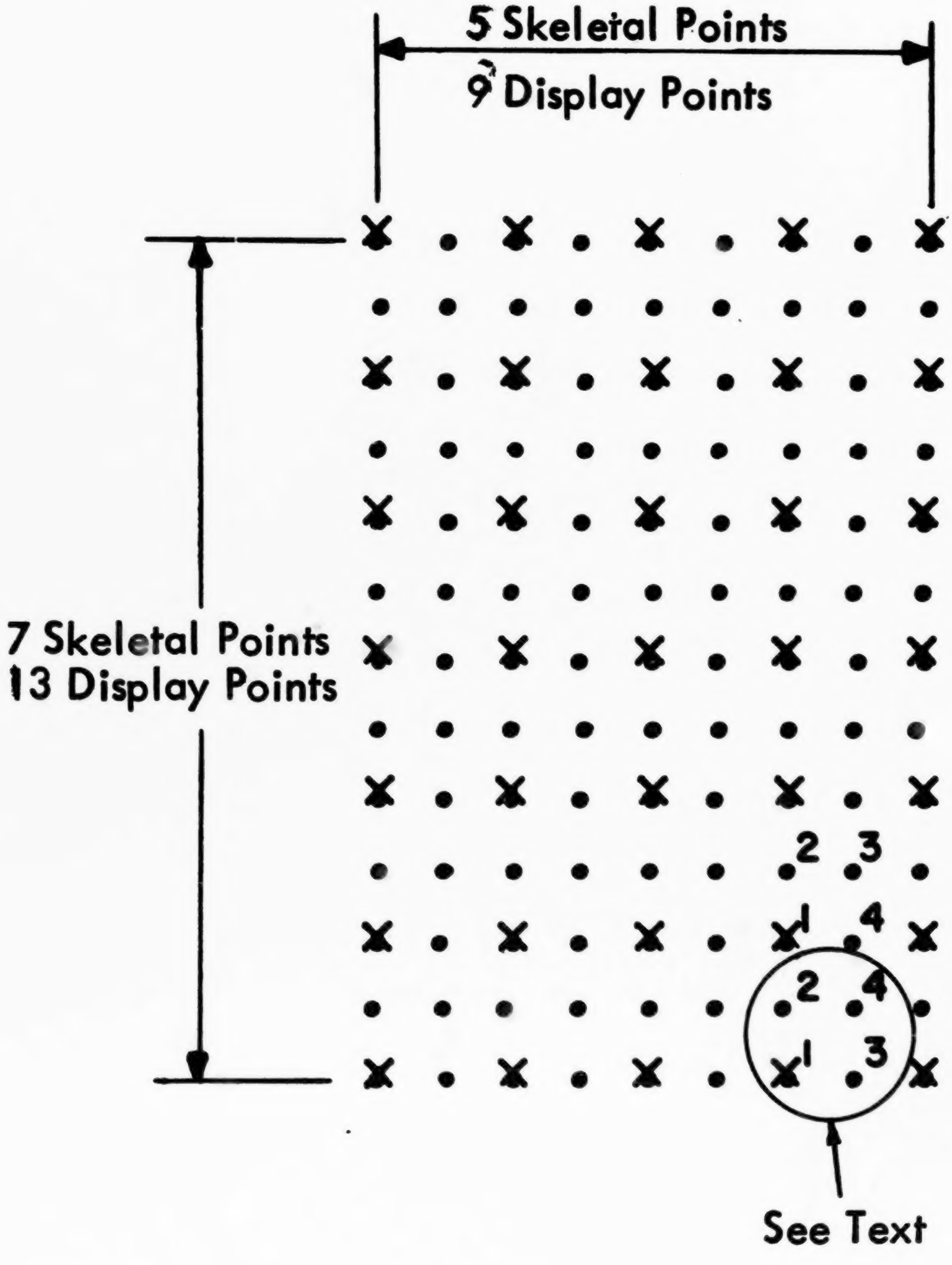
The above observation seemed to suggest a method to economize on the size of the character matrix table. Suppose only the basic shape of the character is stored in a small skeletal matrix. This matrix, if displayed, would look roughly like the desired characters, but because the matrix is small, the character would have many jagged features. It is possible to add additional dots between the skeletal points to make the letter "well rounded". This smoothing occurs just before the matrix is displayed. Thus a much larger matrix can be displayed than is stored in the read-only memory. The memory reduction would be advantageous since Project Intrex is interested in a very large symbol set. The dot matrix for approximately 200 symbols, including upper and lower case Greek and English letters, along with many scientific symbols, will have to be stored in digital memory.

3.3 INTERPOLATION ALGORITHM

The simplest type of expansion from the skeletal matrix is to add a new row (column) between each of the specified rows (columns). Figure 3.3 shows the matrix with skeletal points marked with "x" and displayed points marked with ".". If the stored matrix is n by m, the displayed matrix will be (2n-1) by (2m-1). With large values of n and m, the displayed matrix has roughly four times as many dots as are in the skeletal matrix, leading to a 75% saving in memory.

It is now necessary to specify a smoothing algorithm which works well on characters and also has an economical realization. The following paragraphs attempt to derive the algorithm.

One recognizes a specific shape in a certain area of a character independently of the shape of the rest of the character. To detect the curved bottoms of the letters "O" and "U" for example, a person does not have to look at the top half of the characters. The smoothing algorithm should therefore be able to determine the smoothness of a particular feature of a letter using only the information in a small area of the skeletal matrix.



5 Skeletal Points
9 Display Points
7 Skeletal Points
13 Display Points
2
3
1
4
2
4
1
3
See Text

Fig. 3.3 Type of Display Points

Points Marked X are in the Skeletal Matrix
Points Marked • are in the Display Matrix

The smoothing algorithm will be given only those skeletal points which are within a certain distance of the point being computed. There is a window centered about each point which contains all the information necessary to specify the value of that point.

The curves and diagonals which need smoothing on a character can occur on any area of the character. Thus the type of smoothing is invariant to translation. The smoothing algorithm should use the same rules for smoothing no matter where in the image the window is placed.

It is possible to form a set of all the patterns which can appear in the window around a given point. A certain subset of this set will always cause the given point to be "on". Thus the smoothing consists merely of combinational logic driven by the values of the skeleton points inside the window.

A computer program was written for the PDP-1 computer to test various transformations of the type described. Using the light pen and oscilloscope of the computer, the operator is able to input a skeleton matrix and specify a set of rules for smoothing. The computer applies the smoothing to the skeleton matrix to form the display matrix.

3.4 SIMPLE INTERPOLATION

Two interpolation algorithms were verified by simulation. The second is a refinement of the first. The problem of explaining the smoothing is complicated because there are four classes of display points which must be handled differently. They are:

Type 1: Those which coincide with a skeletal point

Type 2: Those which are one unit above a skeleton point

Type 3: Those which are one unit right of a skeleton point

Type 4: Those which are one unit above and to the right of a skeleton point.

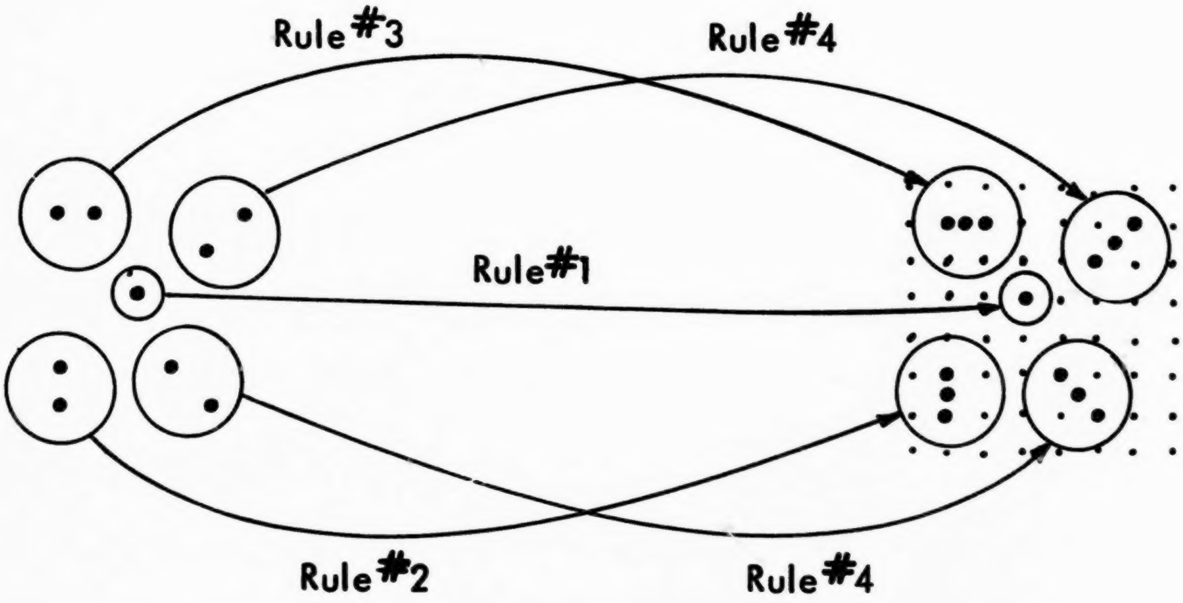
A different combinational logic net must be made for each of them. The more simple type of interpolation can now be described in tabular form. The first column of the table below tells the type of point being described by that line, and the second specifies the condition necessary for that point to be on.

SIMPLE INTERPOLATION

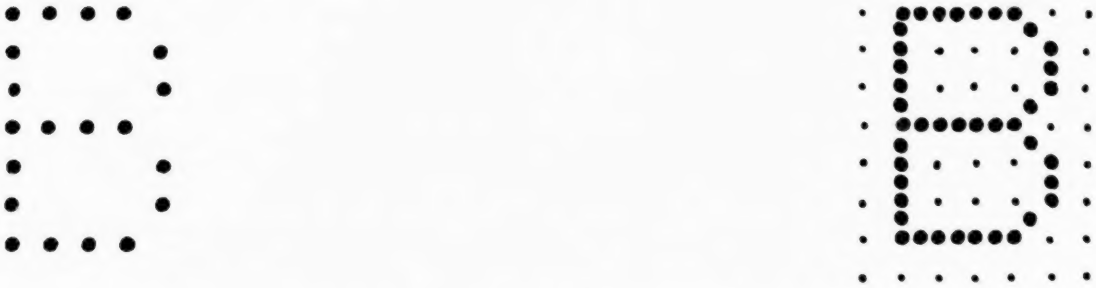
Type point	Condition to be on
Rule 1. (on skeleton)	-corresponding skeleton point is on.
Rule 2. (above skeleton)	-both skeleton points above and below are on.
Rule 3. (right of skeleton)	-both skeleton points to left and right are on.
Rule 4. (right and above)	-two two closest skeletal points on one diagonal are on and the two closest skeleton points on the other diagonal are off.

Each of the four interpolation rules above is illustrated in the simple character in Figure 3.4a. The matrices shown in the left column are skeletal matrices. The matrix to the right of each skeletal matrix is the display matrix which was derived from it.

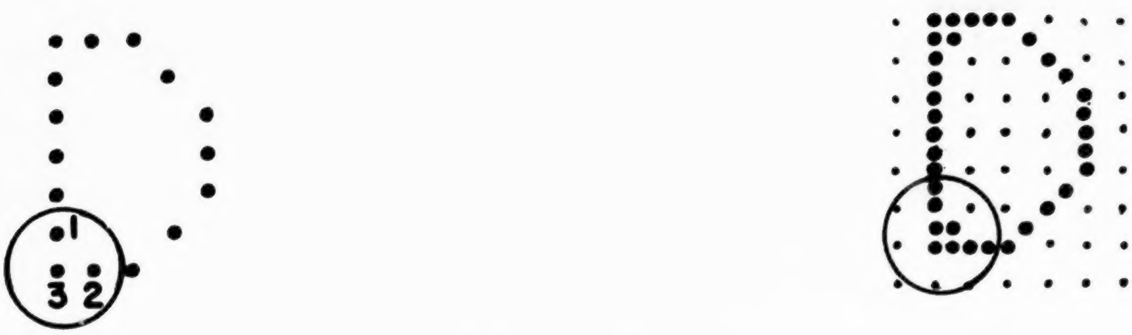
Interpolation rule number 4 (on points of type 4) is composed of two conditions which must both be met. If the second condition (Rule 4, above "...the two closest skeleton points on the other diagonal are off") is not enforced, the resultant interpolation yields characters like the one shown in Figure 3.4c. The problem is that the interpolator is trying to connect points 1 and 2 (see diagram) along the diagonal, and simultaneously along the square-cornered route from points 1 to 3 to 2. It is the job of the second condition in rule 4 to inhibit diagonal interpolation if there is a "square


Rule #3
Rule #4
Rule #1
Rule #2
Rule #4

(a) Simple Interpolation



(b) Filtered Letter



(c) See Text

Fig. 3.4 Simple Interpolation

interpolation" possible. Thus the square corners of characters are not rounded.

3.5 STRONG INTERPOLATION

A second set of interpolation rules is illustrated in the simple patterns of Figure 3.5. This set of rules contains all the rules previously stated plus one more rule:

STRONG INTERPOLATION

Rule 1 through Rule 4 plus:

Rule 5:	If a skeletal point is part of both a diagonal pattern and a vertical (or horizontal) pattern, then that skeletal point is moved to an adjacent display point as shown in Figure 3.5.

The skeletal point mentioned in rule 5 is the point which is at the vertex of the obtuse angle formed by the diagonal pattern and the vertical (or horizontal) pattern. To a certain extent, this point sticks out. The intention of rule 5 is to soften that point's effect by moving it inwards.

Rule 5 helps in many cases, but it also produces the undesirable patterns of lone dots in a row, and square shaped patterns in curved areas. These are evidenced in the characters of Figure 3.6.

The hardware implementation for the two smoothing algorithms described above is indicated in the following chapter, while Chapter 5 discusses their success in forming smooth characters.

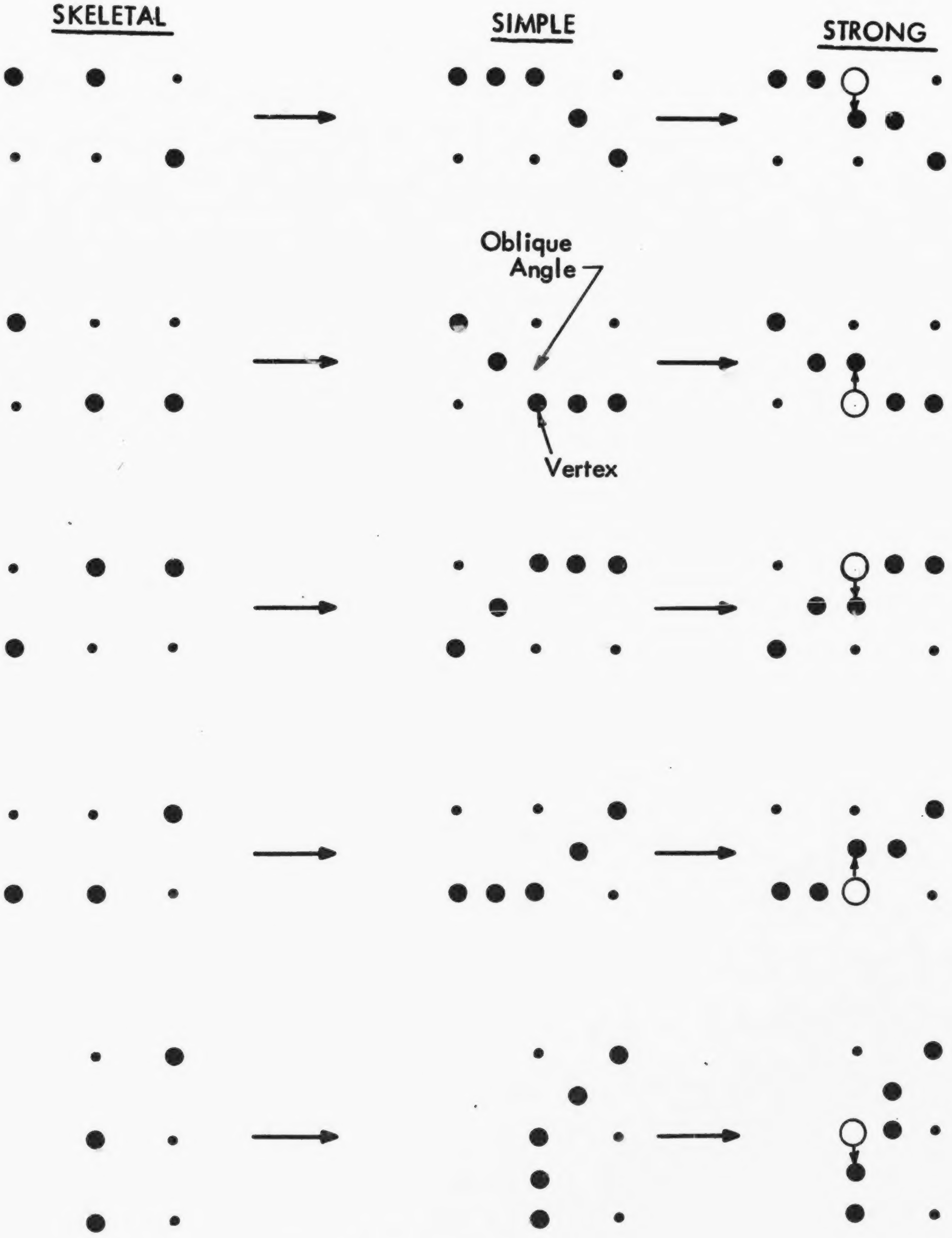

SKELETAL
SIMPLE
STRONG
Oblique Angle
Vertex

Fig. 3.5 Moving Vertex of Oblique Angle

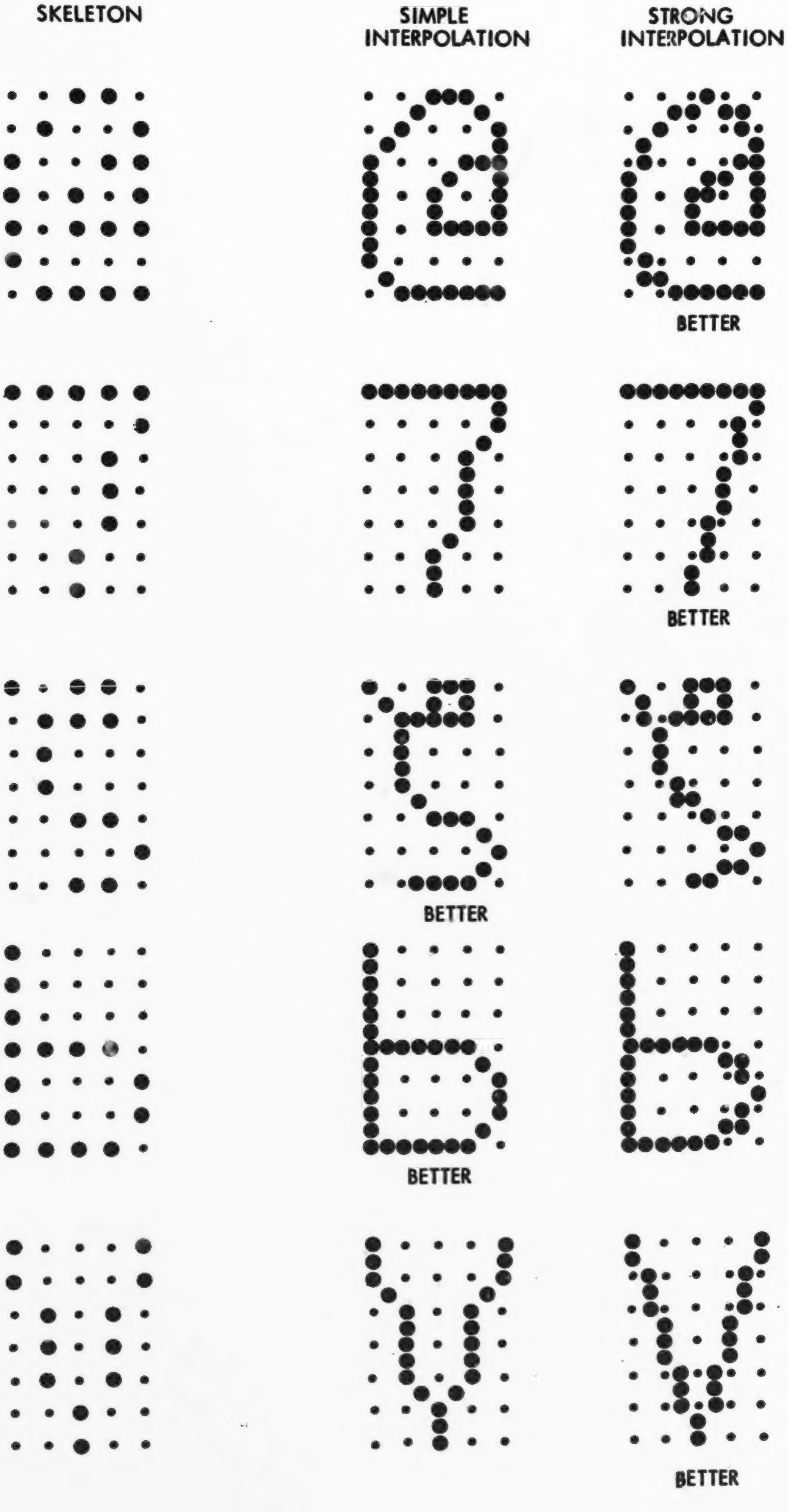

SKELETON
SIMPLE INTERPOLATION
STRONG INTERPOLATION
BETTER
BETTER
BETTER
BETTER
BETTER

Fig. 3.6 Examples of Interpolation

CHAPTER IV

HARDWARE IMPLEMENTATION

Like many other digital systems, the character generator is composed of a section which processes the matrix data, and a control network which directs the data processing. Figure 4.1 shows a block diagram of the whole system. The top half of the diagram is the data section.

The third box in the top row is a commercially available "wire-rope" read-only memory. It stores the skeletal matrix for each of the 192 characters in the character set. Upon receipt of an ASCII code from the console, the read-only memory looks up the 7 by 5 skeletal matrix which corresponds to the code received. Details of the lookup process are discussed in Section 4.3. It sends that matrix to the interpolator, which is represented by the fourth through the sixth box in the top row of Figure 4.1. The interpolator expands the 7 by 5 matrix to a 13 by 9 matrix, using the interpolation algorithm of Chapter 3. While the matrix is being expanded, it is put in the proper format to be displayed using the sinusoidal scan pattern. Since the interpolator is the heart of the thesis, it will be discussed first.

4.1 DATA SECTION

The intensity pattern in the display matrix must be presented to the CRT sequentially in order to be displayed using the sinusoidal scan. This serial nature of the output of the interpolator can be exploited to improve design. To better understand this relation, a series of examples which lead up to the complete data section of the interpolator will be discussed.

4.1.1 SIMPLE "FORMATTOR"

First consider the simple problem of displaying an existing m by n matrix

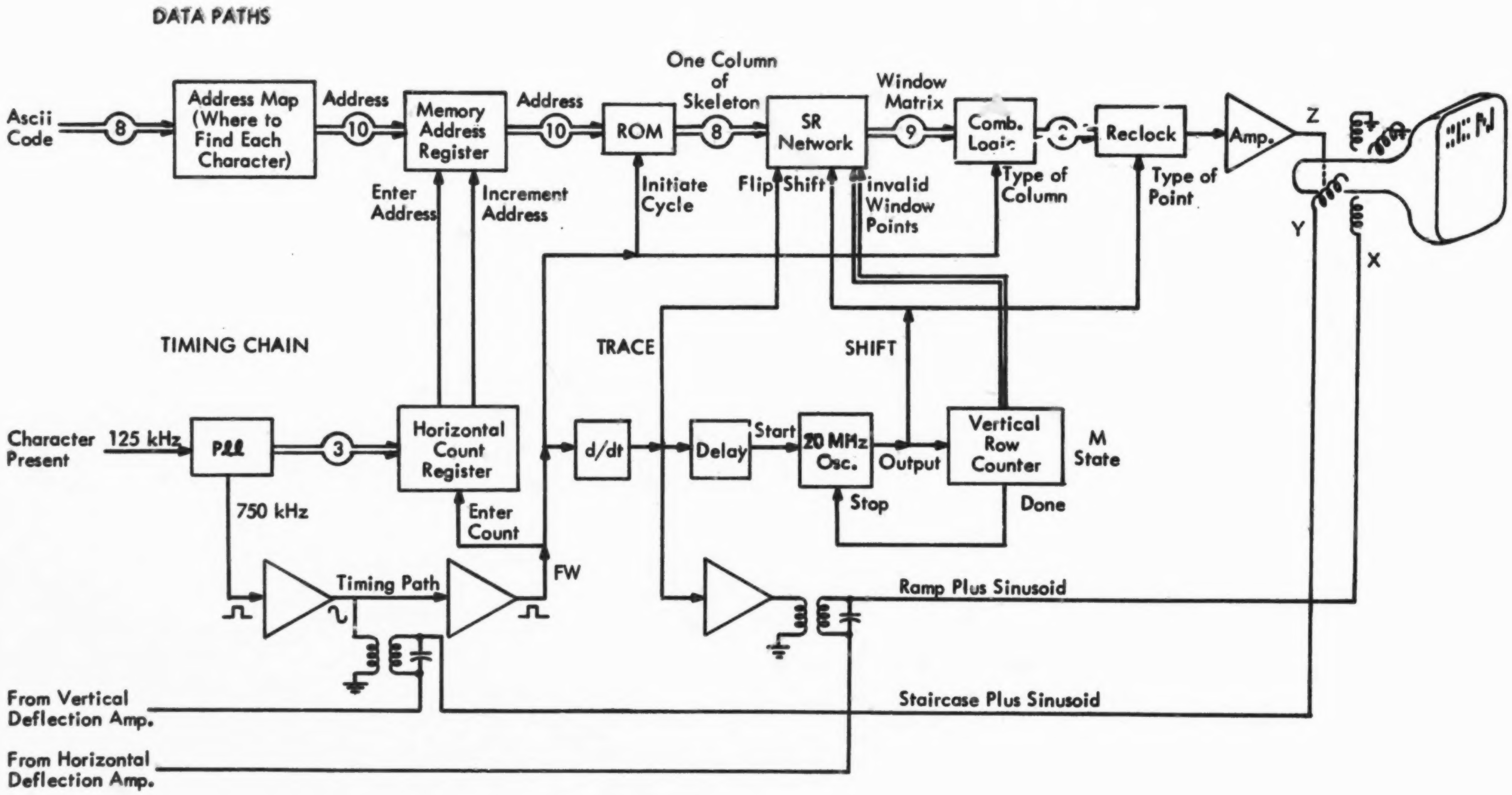

DATA PATHS
Ascii Code
8
Address Map (Where to Find Each Character)
Address
10
Memory Address Register
Address
10
ROM
8
One Column of Skeleton
SR Network
Window Matrix
9
Comb. Logic
Reclock
Amp.
Z
Y
X
Enter Address
Increment Address
Initiate Cycle
Flip
Shift
invalid Window Points
Type of Column
Type of Point
TIMING CHAIN
TRACE
SHIFT
Character Present
125 kHz
PLL
3
Horizontal Count Register
d/dt
Delay
Start
20 MHz Osc.
Output
Vertical Row Counter
M State
750 kHz
Enter Count
Stop
Done
FW
Timing Path
Ramp Plus Sinusoid
Staircase Plus Sinusoid
From Vertical Deflection Amp.
From Horizontal Deflection Amp.

Fig. 4.1 Dot-Matrix Generator Block Diagram

without interpolation. Suppose the matrix is divided up into n m-bit words which are presented to the formattor one word at a time. To simplify matters, assume for the moment that only every other display region is being used to display dots. In this case all scans used for displaying are in the same direction. For purposes of discussion this direction will be considered down the columns of the matrix.

The formattor described here is nothing more than a parallel to serial converter. The converter accepts one m-bit word of the matrix at a time. It converts that word to a serial string before accepting the next word. Two possible implementations exist. The first is an m-position digitally controlled switch. (Figure 4.2 shows a 4-position switch.) When the control bits form a binary k, the k'th input is presented at the output. The control bits are generated by a counter, sequenced from 0 to m-1 by a clock which occurs whenever another bit of the serial output is desired. The second conversion method uses a parallel-in, serial-out shift-register. (Figure 4.3 shows a four-bit shift-register converter.) First, the information is transferred in parallel into the m bits of the register. Then the m bits are serially shifted out of the register by the output clock. The "switching" method is less complex, but the shift-register method has the advantage that it can be used as a memory unit also. Both methods will be used in the final interpolator.

4.1.2 COLUMN INTERPOLATION

Consider augmenting the formattor to do a simple type of interpolation described in interpolation rules no. 1 and no. 2 of Chapter 3. A new dot will be added by the interpolator directly after each dot of the row being formatted. Two values are generated on one output line for each

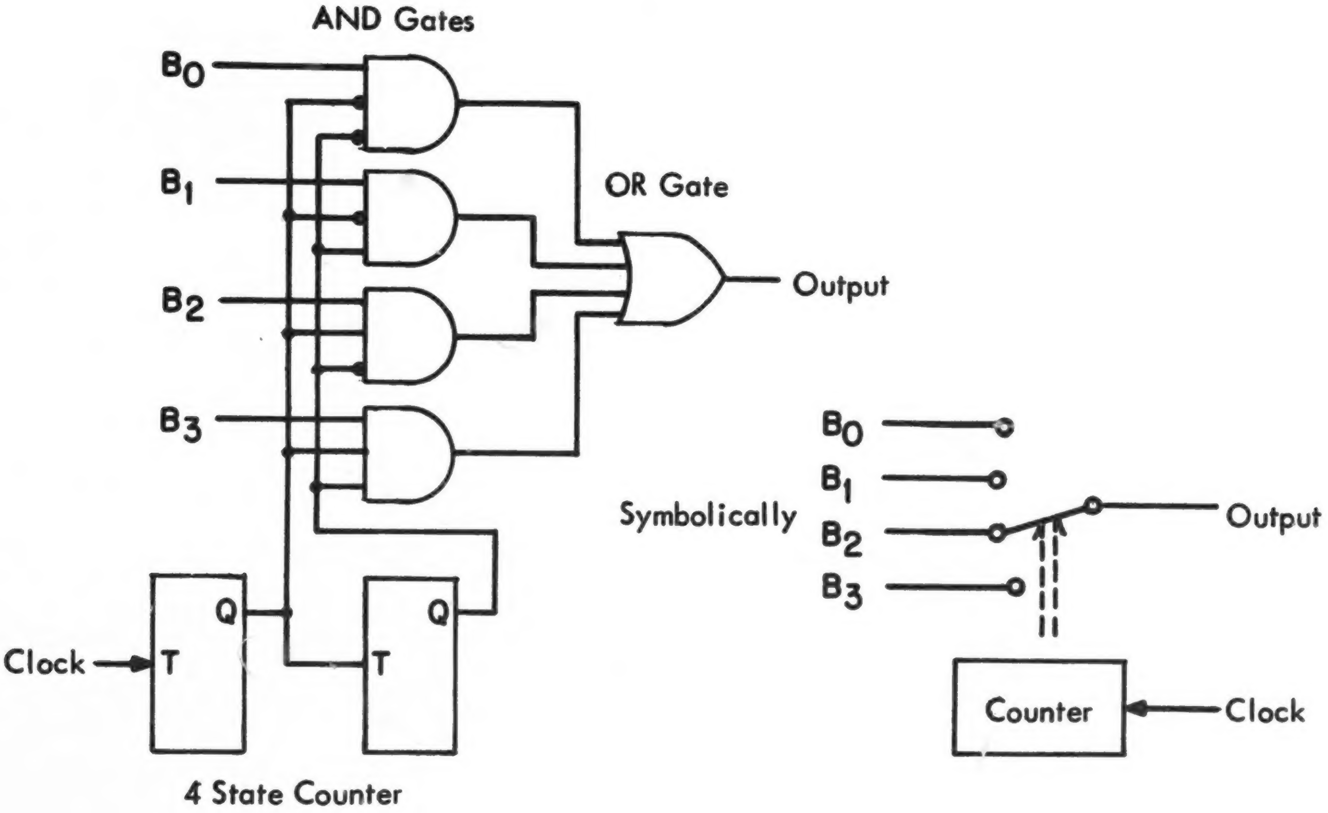

AND Gates
B_0
B_1
B_2
B_3
OR Gate
Output
Q
T
Q
T
Clock
4 State Counter
Symbolically
B_0
B_1
B_2
B_3
Output
Counter
Clock

Fig. 4.2 Digitally Controlled Switch

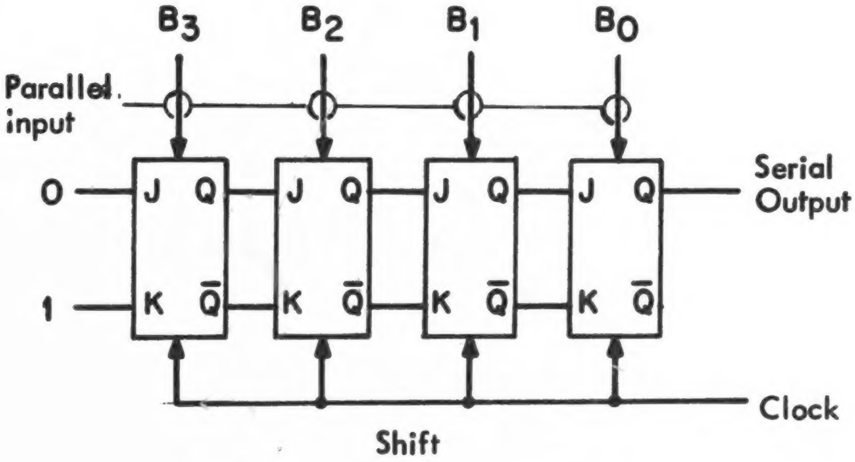

B_3
B_2
B_1
B_0
Parallel input
0
1
J
Q
K
$\bar{Q}$
Serial Output
Clock
Shift

Fig. 4.3 Shift-Register Converter

bit B_k. They are presented one after another, one at time k and the other at time $k + \frac{1}{2}$. A mathematical description of the output follows:

$$\text{Output } (k) = B_k$$
$$\text{Output } (k + \tfrac{1}{2}) = B_{k+1} \wedge B_k$$

These rules cause "display points" to be added if they are between two "skeletal points". Figure 4.4 shows a matrix under this interpolation.

Although this interpolator could be implemented using two digitally controlled switches, it is much simpler to design it using a shift-register converter as shown in Figure 4.5. The converter presents the last bit of the shift-register when the signal PRIMARY DOT is true, and it presents the logical AND of the last bit and the next to the last bit when the signal SECONDARY DOT is true. The shift register is shifted on the trailing edge of SECONDARY DOT.

4.1.3 SIMPLE INTERPOLATION

Interpolation rules no. 1 and no. 2 have been effectively implemented. Additional problems are encountered in adding rules no. 3 and no. 4, which allow for adding dots in horizontal or diagonal rows. The extra dots generated by these rules all fall into an extra column which must be added between the columns of the skeletal matrix. The interpolator must generate this "secondary row" after it generates each "primary row" described by the first type of interpolator. Another problem is that these rules involve the comparison of points in two adjacent rows. Both rows must be present in the interpolator at the same time.

One way to have both rows present is to add another m-bit shift register as shown in Figure 4.6. This register is used to hold the row previously processed. When the interpolator is generating a primary row, the input B_k's are converted to signals on the lines W_{11} and W_{21}. They

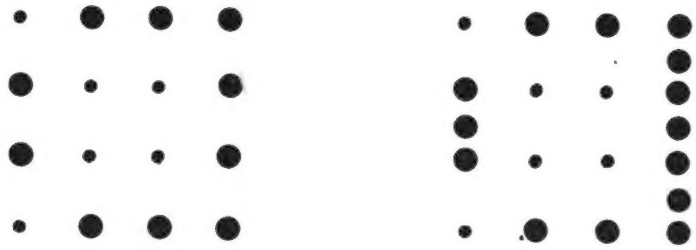


Fig. 4.4 Effect of Rule#1 and#2

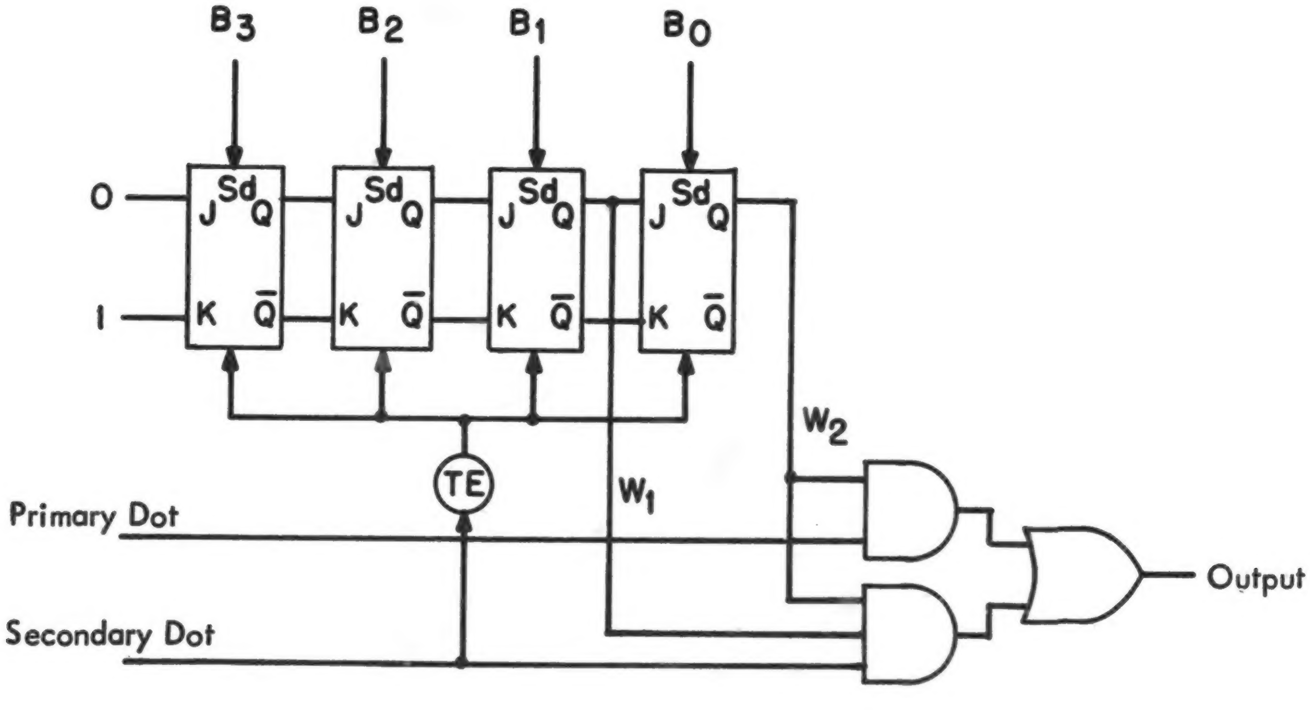

B3
B2
B1
B0
0
1
J
Sd
Q
K
$\bar{Q}$
TE
W2
W1
Primary Dot
Secondary Dot
Output

Fig. 4.5 Interpolator for Rule#1 and#2

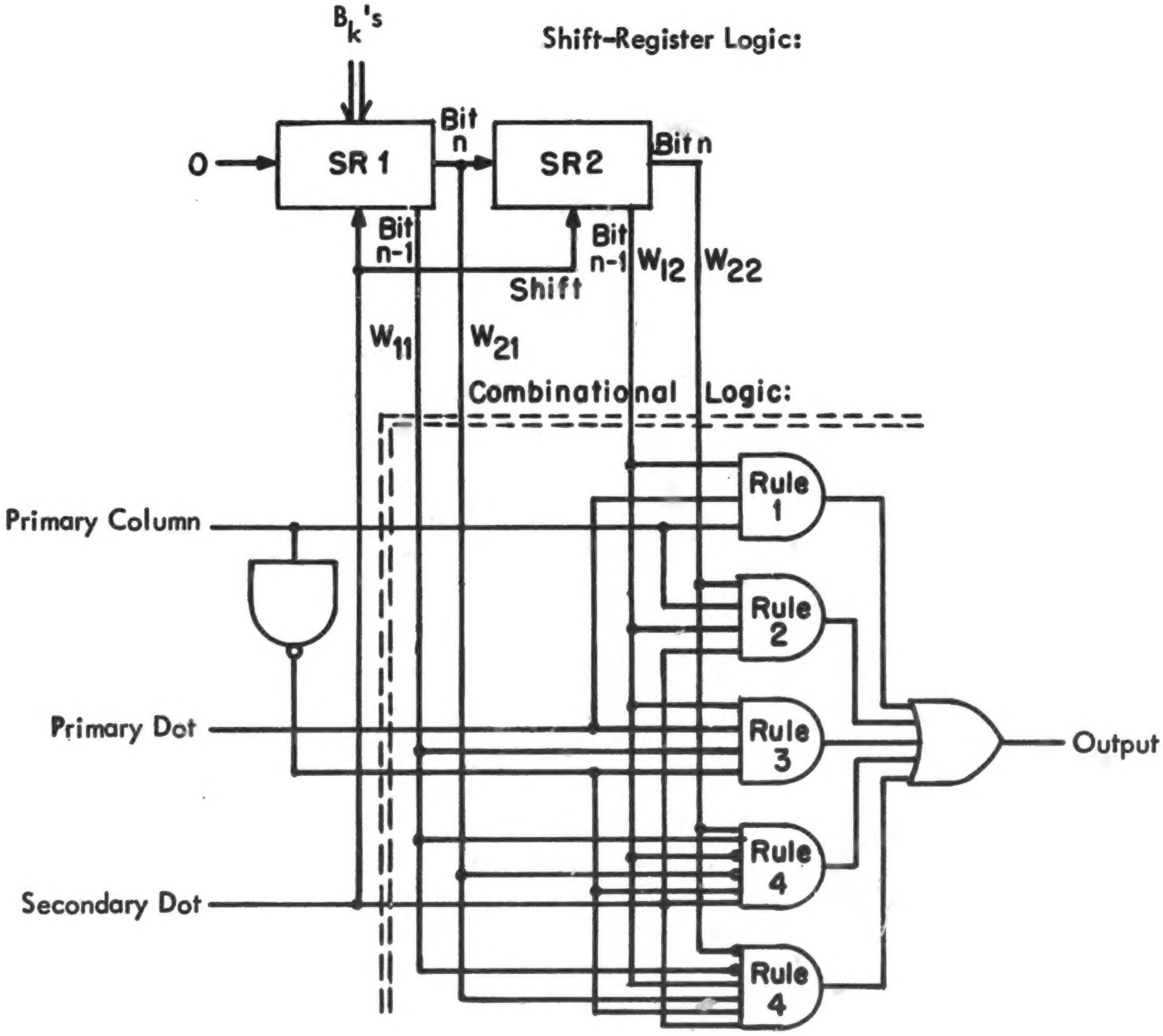

B_k's
Shift-Register Logic:
0
SR 1
Bit n
SR2
Bit n
Bit n-1
Bit n-1
Shift
W_{12}
W_{22}
W_{11}
W_{21}
Combinational Logic:
Primary Column
Rule 1
Rule 2
Primary Dot
Rule 3
Output
Rule 4
Secondary Dot
Rule 4

Fig. 4.6 Interpolator for Rules #1 Through #4

are used as before to perform type no. 1 and no. 2 interpolation. While this interpolation is being performed, shift register 2 is being loaded with the serial output of shift register 1.

When the interpolator is generating a secondary column, a new word of B_k's are converted to signals on lines W_{11} and W_{21}. In addition, W_{12} and W_{22} scan the values of the previous column in the same manner that W_{11} and W_{12} do for the new columns.

The outputs W_{11}, W_{12}, W_{21}, W_{22} are named with subscripts because they can be thought of as a matrix. This matrix is actually the window matrix centered about the dot being generated. The shift register structure described in Figure 4.6 causes the window to slide down the rows of the matrix covering two columns at a time. At any instant, the window contains all the information necessary to do the desired interpolation. The values in the window, along with the type of row (primary or secondary) and type of point (primary or secondary) are fed to a combinational logic net to determine whether that point is on.

An example may be helpful at this time. Figure 4.7a shows a skeletal matrix, and Figure 4.7b shows the corresponding interpolated matrix. Figure 4.7c shows the contents of the two shift registers as the first two columns of the interpolated matrix in Figure 4.7b are being generated.

The first entry in Figure 4.7c labeled "example", shows the two shift registers containing the arbitrary literals A,B,...J. The last two outputs of SR1, (D and E) and SR2 (I and J) are used to form the window matrix. In the columns labeled "Primary" and "Secondary", the literals, A,B,...J have been replaced by specific values 0, 1, or $\emptyset$. The symbol "$\emptyset$" represents the don't care condition: the interpolation is independent of its value. The purpose of the literals is to show the relation between the bits of the

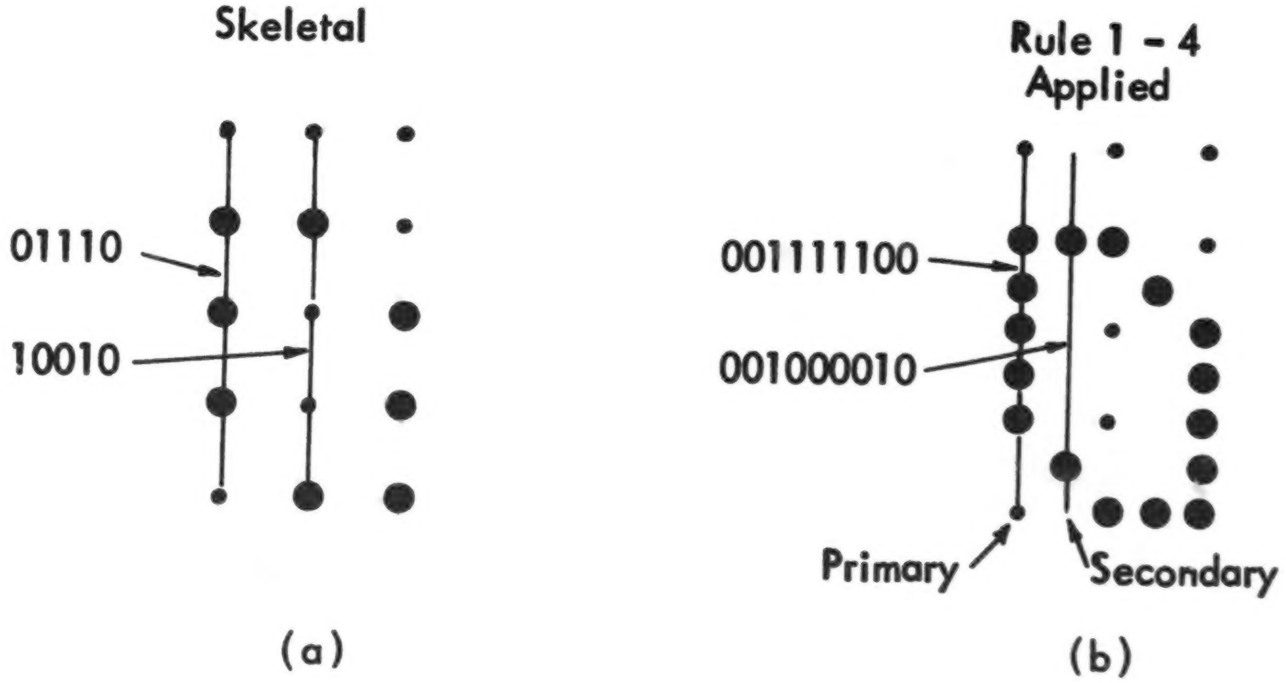

Skeletal
01110
10010
(a)
Rule 1 - 4
Applied
001111100
001000010
Primary
Secondary
(b)

Fig. 4.7 Effect of Rule#1 - Rule#4

Type column	Contents of SR1	Contents of SR2	Window matrix	Output time	Output value	Rule used
example:						
	ABCDE	FGHIJ	JE ID			
	load word					
Primary:						
	01110	øøøøø	ø0 ø1	1.0	0	-
				1.5	0	-
	ø0111	0øøøø	ø1 ø1	2.0	1	#1
				2.5	1	#2
	øø011	10øøø	ø1 ø1	3.0	1	#1
				3.5	1	#2
	øøø01	110øø	ø1 ø0	4.0	1	#1
				4.5	0	-
	øøøø0	1110ø	ø0 0ø	5.0	0	-
	øøøøø	01110				
	load new word					
Secondary:						
	10010	01110	00 11	6.0	0	-
				6.5	0	-
	ø1001	00111	11 10	7.0	1	#3
				7.5	0	-
	øø100	10011	10 10	8.0	0	-
				8.5	0	-
	øøø10	01001	10 01	9.0	0	-
				9.5	1	#4
	øøøø1	00100	01 0ø	10.0	0	-
	øøøøø	01001				

Fig. 4.7c Contents of Shift Registers

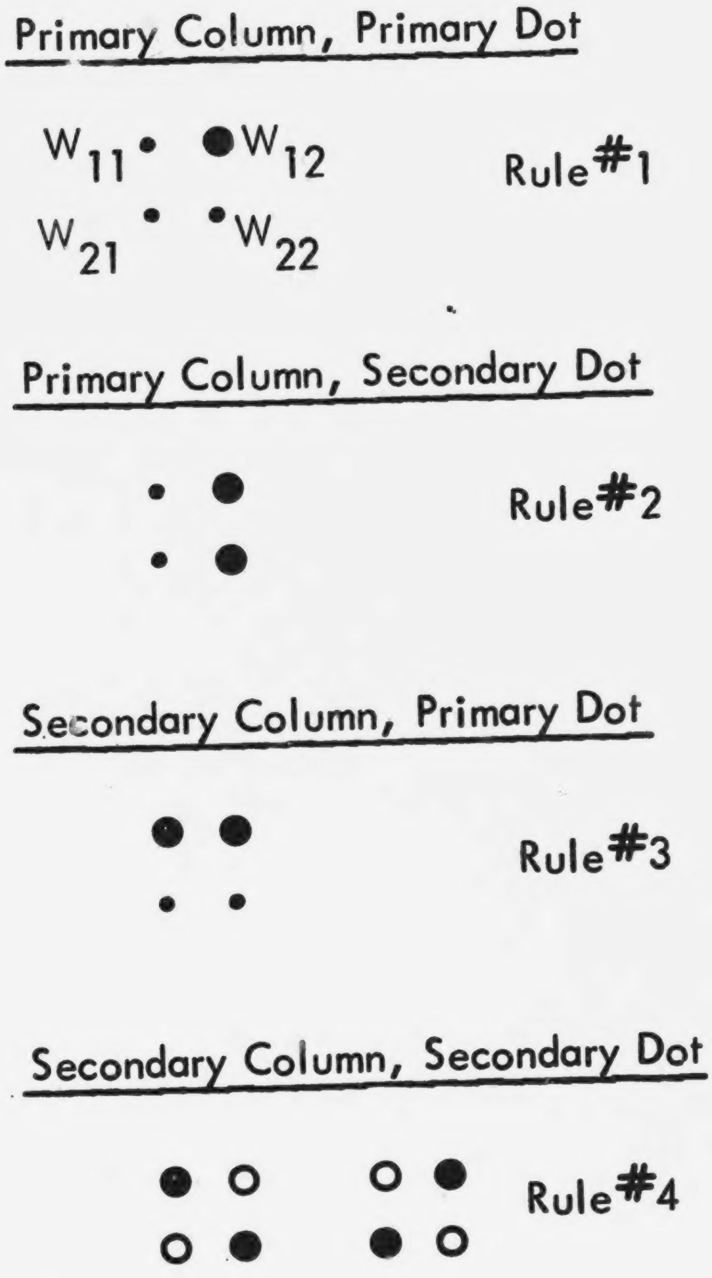

Primary Column, Primary Dot
W_{11}
W_{12}
W_{21}
W_{22}
Rule#1
Primary Column, Secondary Dot
Rule#2
Secondary Column, Primary Dot
Rule#3
Secondary Column, Secondary Dot
Rule#4

Fig. 4.7d Patterns in the Window Matrix for Simple Interpolator

shift registers and the bits of the window matrix.

Before output time 1.0, the binary word 01110 (the first column of Figure 4.7a)is entered into SR1. As the primary scan progresses, this word is shifted into SR2. In the process, consecutive pairs of bits in the word are present in the window matrix.

Figure 4.7d shows the patterns which must be present in the window matrix if the output value is to be a 1. A heavy dot indicates that the bit must be a 1, a circle indicates that the bit must be a 0, and a light dot indicates that the value of that bit has no effect on the decision. Given that a certain type of dot (primary or secondary) is to be generated in a certain type of column (primary or secondary), one can look in the appropriate portion of Figure 4.7d. If there is a pattern in that portion which matches the pattern in the window matrix, then the output of the interpolator is a 1. In this manner the binary string 001111100 is produced by the primary scan. This is the first column in Figure 4.7b. The last column in Figure 4.7c shows which rules were used to generate each 1 in the string.

After the primary scan, a new word (binary 10010) is loaded into SR1. This is the second column of Figure 4.7a. Another scan is performed except this time the rules for interpolation are for a secondary row. The string produced, 001000010, is the second column of Figure 4.7b.

4.1.4 ORDER REVERSING

It is possible to remove the restriction that only scans down a column are used for displaying dots. Dots can be displayed in every display region, whether the beam is scanning upward or downward. The problem encountered is that the display matrix points must be generated in reverse order when the beam is scanning upward. Since the interpolation rules operate equally well on matrices whose columns are upside down, the column can be generated in reverse order, by sliding down a skeletal matrix whose columns have been turned upside down. Thus all one needs to do is to insure that the matrix is upside down when generating an upward column. The interpolation can then be carried out in the same manner as a downward column.

It is necessary to have the matrix in its upside down form every odd scan, and in its right side up form every even scan. The contents of the shift registers must therefore have their order reversed between each scan. Bit 1 is swapped with Bit n, Bit 2 with Bit n-1, etc. This feat can be accomplished in one parallel operation as shown in Figure 4.8. When shifting, information is entered into each cell through its input A. When flipping, information is entered through its input B. (An additional parallel entry path for the B_k's has not been shown because it is not needed in the next section.)

An alternative approach to the problem is to have bi-directional shift registers. The registers would shift forward during one scan, and backward during the next scan. Bi-directional shift registers and order flipping shift registers in themselves require the same logic to implement. Formation of the window matrix using bi-directional shift registers is more complicated because different bits of the shift register must be used when shifting in different directions. They were not employed for that reason.

4.1.5 STRONG INTERPOLATION

This is the final example in the series of machines: this is the description of the machine which was built for the Intrex console. Interpolation rules no. 1 through no. 5, which do all the previous filtering and, in addition, move the vertex of an obtuse angle inwards, can now be implemented. These rules require a 3 by 3 window matrix for generation of the primary columns, and a 3 by 2 matrix for the generation of secondary columns.

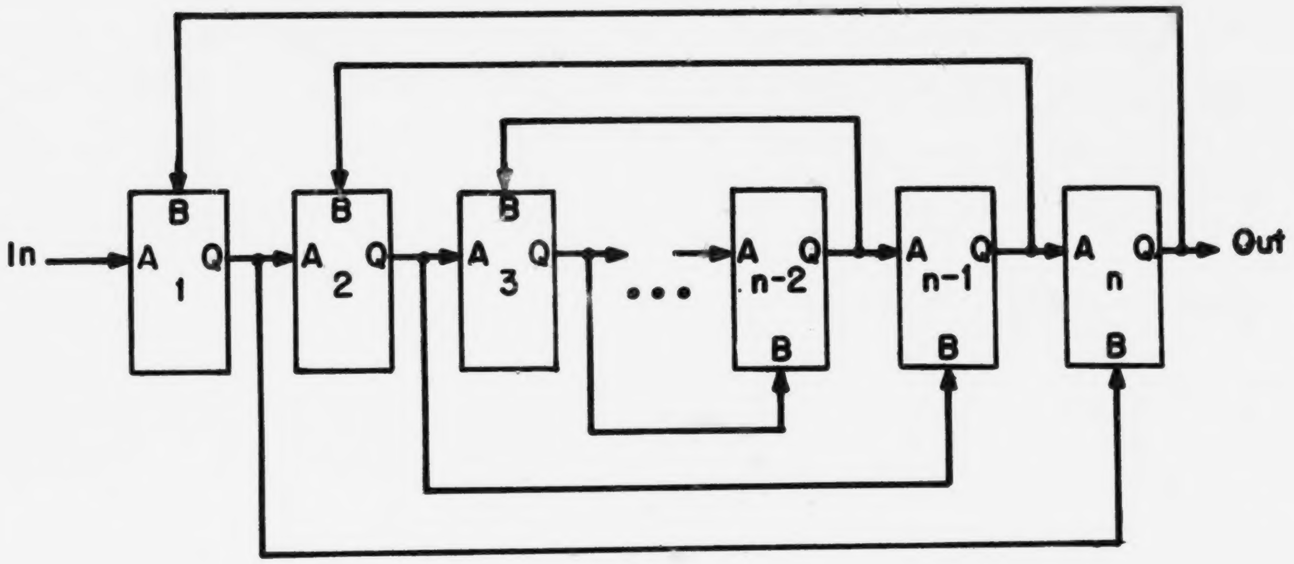

In
A
B
Q
1
2
3
...
n-2
n-1
n
Out

Fig. 4.8 Bit-Reversing Shift Register

This requirement will be substantiated in a few pages.

It is necessary to form a configuration of shift registers which will produce the larger window matrix. The goal is basically an extension of the 2 by 2 configuration. To obtain an extra column in the window matrix, another shift register is added. To obtain an extra row, one clocked delay is added and one more output is taken from each shift register. In addition, part of SR1 has been replaced by a digitally controlled switch to reduce logic. The data structure, shown in Figure 4.9, is a little more complicated, and is described in the following paragraphs.

During the primary scan, one m-bit word is presented in parallel on lines B_k. Figure 4.10 defines the process explicitly. If the column of the display matrix corresponding to column i in the skeletal matrix is about to be generated, then column i+1 is present on the B_k's at this time. This advance allows the interpolator to look one column ahead of the column it is generating. SR2 is assumed to contain column i, and SR3 is assumed to contain column i-1. These assumptions will be justified shortly.

Before the scan is started, B_1 is loaded into one of the one bit delays shown in Figure 4.9. Before shifting is started, W_{21}, W_{22}, and W_{23} are the first bits in column i+1, i, and i-1 respectively of the skeletal matrix. W_{31}, W_{32} and W_{33} contain the second bit of their respective columns. W_{11}, W_{12}, and W_{13} contain incorrect values. This error corresponds to the situation depicted in Figure 4.11. Part of the large window matrix falls outside the skeletal matrix into an undefined area. To prevent any points from being incorrectly added to the interpolator's output, these values are defined to be zero during the first shift state. The AND gates in Figure 4.9 perform this function.

During the primary scan, the data switches are set so that SR2 is

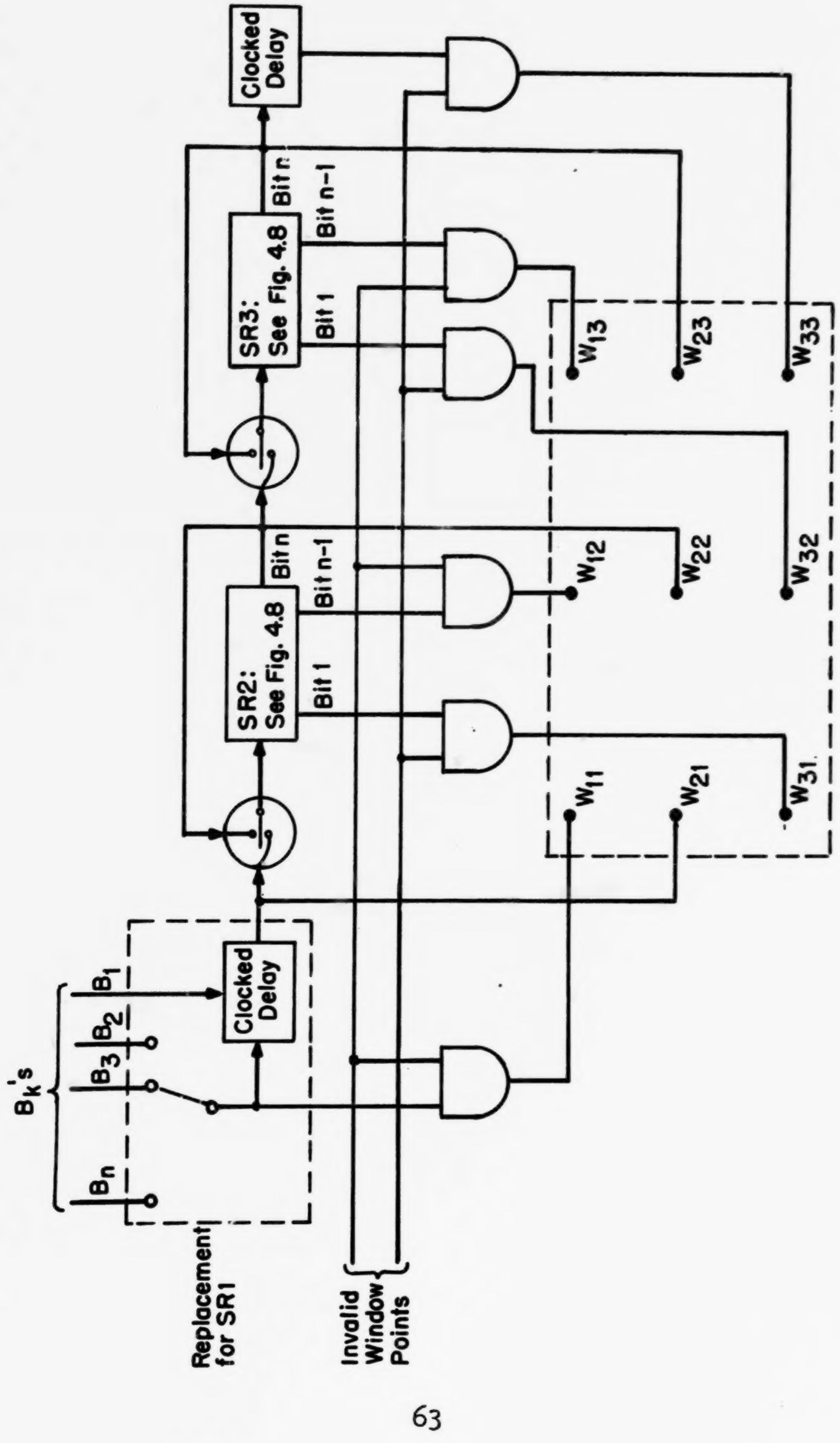

B$_k$'s
B$_n$
B$_3$
B$_2$
B$_1$
Clocked Delay
Replacement for SR1
Invalid Window Points
SR2: See Fig. 4.8
Bit 1
Bit n-1
Bit n
SR3: See Fig. 4.8
Bit 1
Bit n-1
Bit n
Clocked Delay
W$_{11}$
W$_{21}$
W$_{31}$
W$_{12}$
W$_{22}$
W$_{32}$
W$_{13}$
W$_{23}$
W$_{33}$

Fig. 4.9 Shift-Register Network for Rules #1 Through #5

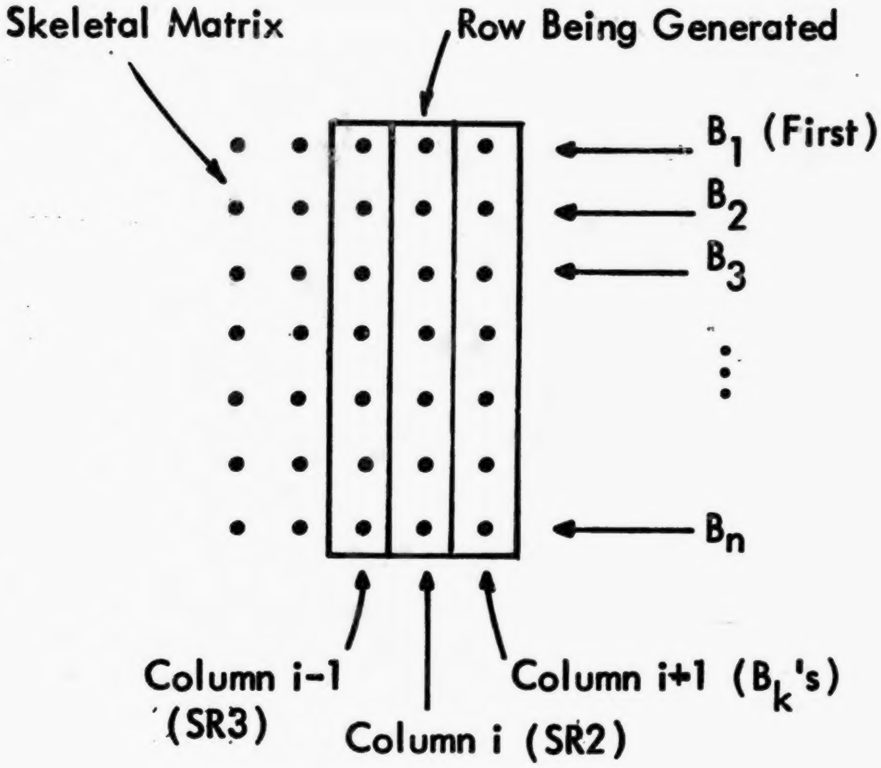

Skeletal Matrix
Row Being Generated
B_1 (First)
B_2
B_3
B_n
Column i-1 (SR3)
Column i (SR2)
Column i+1 (B_k's)

Fig. 4.10 Data Present in Interpolator

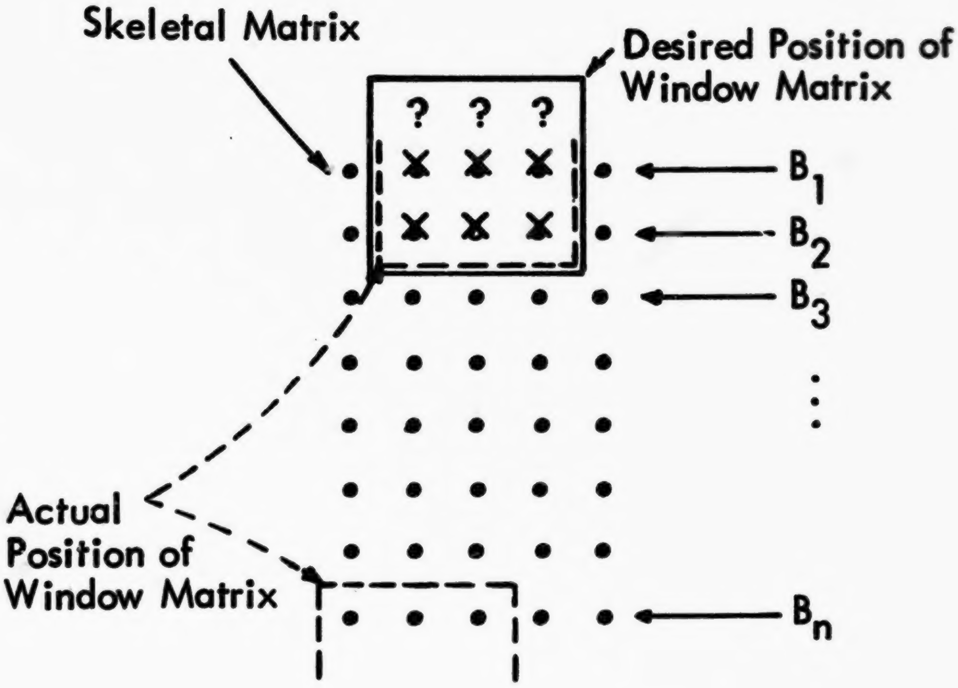

Skeletal Matrix
Desired Position of Window Matrix
? ? ?
× × ×
× × ×
B_1
B_2
B_3
B_n
Actual Position of Window Matrix

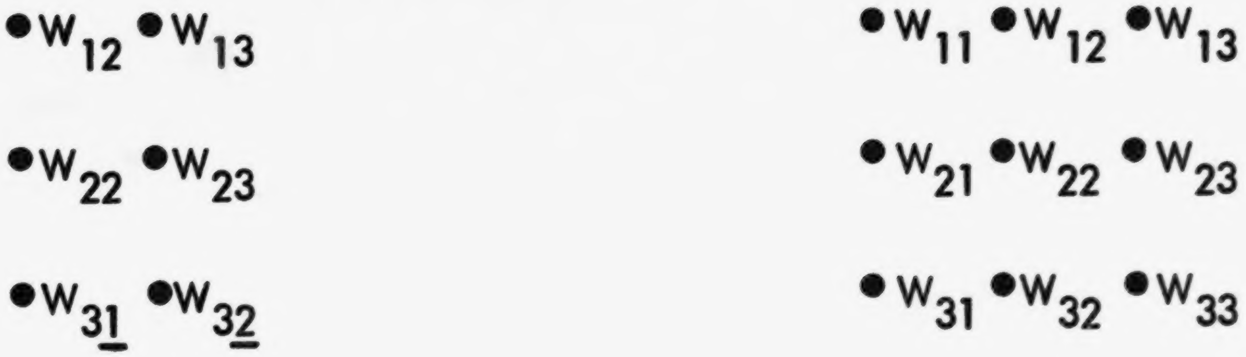
Fig. 4.11 Undefined Values in Window Matrix

shifted into SR3 and the B_k's are converted to a serial string and shifted into SR2. The clocked delay is always driven by the output of SR3. It is used to generate W_{33}. Once the registers have been shifted once, all the windows contain correct values. For example, W_{31} contains bit 1 of column i+1, W_{21} contains bit 2 of that column, and W_{11} contains bit 3. After each shift-pulse, the bits advance in the shift registers, and the window appears to slide down the columns of the skeleton matrix. Until the last shift pulse, all the values of the window matrix are valid. Then part of the window matrix extends below the skeletal matrix. The remedy is again to AND out the invalid levels during the last shift time.

After the primary scan is completed, SR2 contains column i+1, and SR3 contains word i. Now the interpolator is ready to generate a secondary column between column i and column i+1. Since this column is displayed on the upswing of the scan, it must be generated in the opposite direction. The shift registers must have their order reversed at this time.

During the secondary scan, the data flow switches cause the shift-registers to recirculate upon themselves. The undefined conditions caused by the window matrix extending outside the skeletal matrix exist in this scan as before and are treated using the same logic.

Because only a 3 by 2 window matrix is needed to do this secondary interpolation, only two shift-registers, SR2 and SR3, take an active part in producing the window matrix at this time. The digitally controlled switch is not used during this scan. The 3 by 2 window matrix is shown in Figure 4.12. Unfortunately, the indices of the W's do not coincide with their position in the window matrix. This is only a conceptual problem - the combinational logic can be wired around it.


W_{12} W_{13}
W_{22} W_{23}
$W_{3\underline{1}}$ $W_{3\underline{2}}$
W_{11} W_{12} W_{13}
W_{21} W_{22} W_{23}
W_{31} W_{32} W_{33}

Secondary Scan Primary Scan

Fig. 4.12 Window Matrices Definition

After the secondary scan is completed, SR2 contains word i+1, and SR3 contains word i. It is now necessary to re-flip the bits of the shift-registers back to their forward order, since the next scan is in the forward direction. The net effect of advancing and flipping the columns during the primary scan, and recirculating and flipping them during the secondary scan is to put SR2 into SR3 and to put the B_k's into SR2, both in unreversed form. It is easy to see that after a few cycles SR2 will always contain the previous column processed, and SR3 will contain the column processed before that. Thus our original assumption to that effect was justified.

Figure 4.13 shows the hardware logic diagram which performs all the shifting and flipping operations necessary to generate the window matrix.

Figure 4.14 shows the patterns which must appear in the window to generate the four types of points in the display matrix. The combinational logic in Figure 4.15 implements these rules.

Two additional features are shown in that figure. It is possible to either apply rule no. 5 or not. If the level labeled FILTER MODE 1 is TRUE, rules no. 1 through no. 5 are used to generate the display matrix. (Strong interpolation is performed.) If FILTER MODE 1 is FALSE, only rules no. 1 through no. 4 are used. (Simple interpolation is performed.) This provision was added because best results are obtained by interpolating some characters using simple interpolation, and others using strong interpolation. An extra bit is stored in the read-only memory for each character, to tell which type of interpolation is to be applied.

The second addition was required because of the high operating frequency of the interpolator. Since a dot is generated every 25 nanoseconds, races in the combinational logic become critical. Note that strings of func-

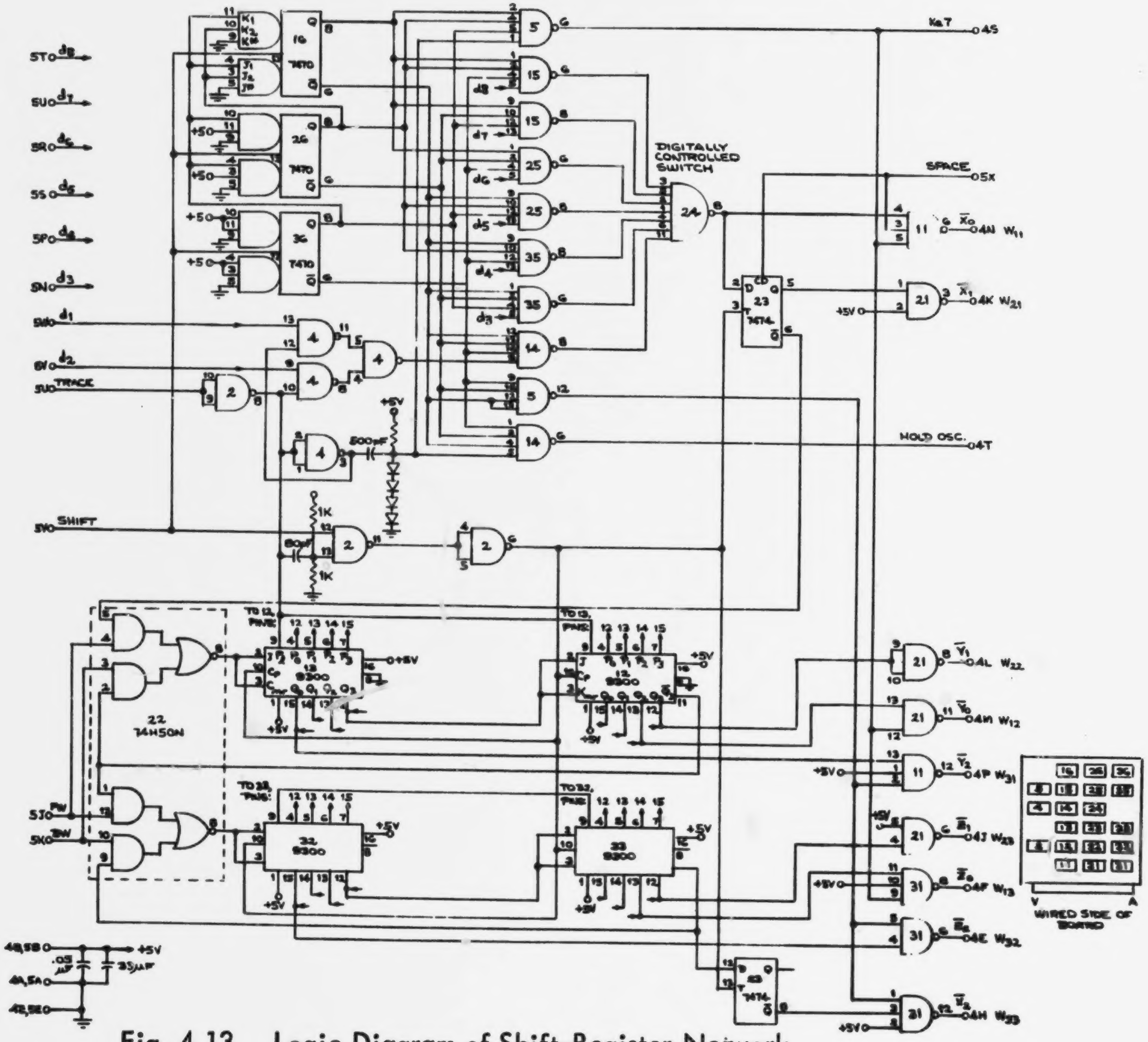
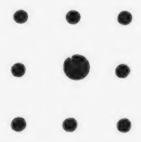
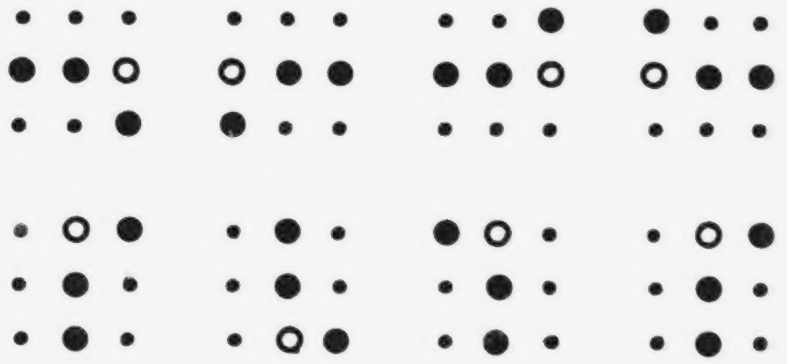

ST0 d8
SU0 d7
SR0 d6
SS0 d5
SP0 d4
SN0 d3
SW0 d1
6V0 d2
SU0 TRACE
5Y0 SHIFT
5J0 PW
5K0 SW
7470
DIGITALLY CONTROLLED SWITCH
K∙T 045
SPACE 05X
HOLD OSC. 04T
04N W11
04K W21
7474
500pF
1K
1K
80pF
+5V
22 74H50N
TO 12 PINS
TO 13 PINS
TO 32 PINS
TO 33 PINS
9300
04L W22
04M W12
04P W31
04J W23
04F W13
04E W32
04H W33
WIRED SIDE OF BOARD
48,5B0
44,5A0
42,5E0
.05 μF
25μF

Fig. 4.13 Logic Diagram of Shift-Register Network

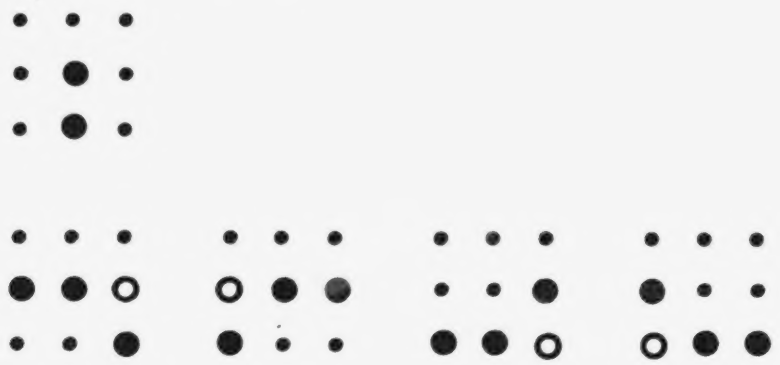
Primary Column, **Primary Dot** is on only if the Pattern



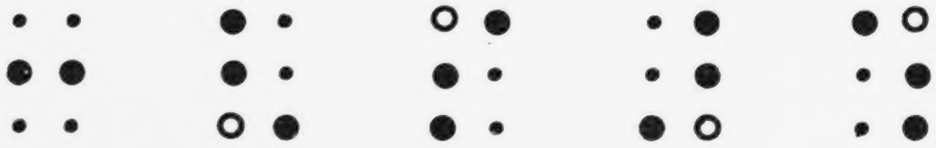
is Present in the Three by Three, and None of the Following Patterns are Present



Primary Column, **Secondary Dot**



Secondary Column, **Primary Dot**



Secondary Column, **Secondary Dot**

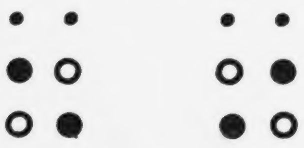


Fig. 4.14 Patterns in the Window Matrix for Strong Interpolation

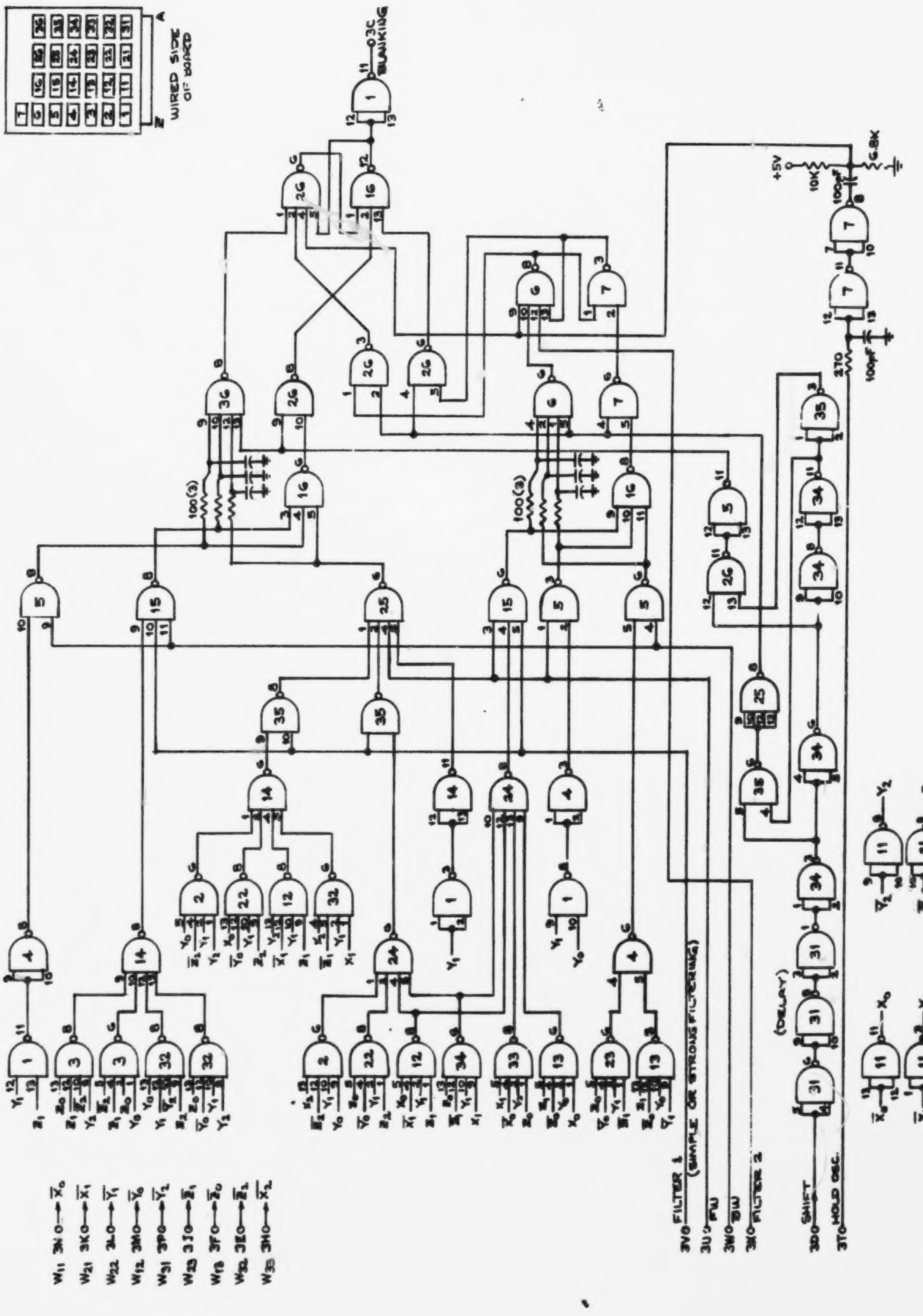

WIRED SIDE OF BOARD
03C BLANKING
FILTER 1
(SIMPLE OR STRONG FILTERING)
FW
BW
FILTER 2
SHIFT
HOLD OSC.
(DELAY)
+5V
10K
6.8K
100pF
270
100(3)
W11 3N0 → X̄0
W21 3K0 → X̄1
W22 3L0 → Ȳ1
W12 3M0 → Ȳ0
W31 3P0 → Ȳ2
W23 3J0 → Z̄1
W13 3F0 → Z̄0
W32 3E0 → Z̄2
W33 3H0 → X̄2

Fig. 4.15 Interpolation Logic Diagram

tionally useless inverters or R-C delays have been added to several paths. This forces all paths to be within about one gate delay of each other. To prevent any remaining races from reaching the CRT, the two outputs for each shift-pulse are stored in two buffer flip-flops before they are multiplexed onto one output line.

The blanking amplifier, shown in Figure 4.16, was designed by Professor J. K. Roberge. It is capable of driving a 30 volt waveform into a 20pF load with a rise time of 6ns. The output stage is a complementary common emitter amplifier.

4.2 TIMING CHAIN

The timing chain directs the operation of the data section of the interpolator just described. Supplied with only a single pulse every character, the timing chain forms some 200-odd time periods which are necessary to display that character. Design was complicated by the following facts:

1) The timing of the unit must follow the timing of the drum. The unit cannot operate asynchronously.
2) A periodic signal must be used to produce the sinusoidal scan, and the appropriate frequency is not recorded on the drum.
3) Delays in the deflection circuits must be compensated for by the timing circuitry.

It proved impossible to synchronize all timing signals with one high speed clock. Pulses are consequently formed by subdividing the pulses from the drum into many shorter ones. The horizontal generator (Section 4.2.1) produces a fixed number of periodic column scans for each character pulse from the drum. One character-time interval is thus divided into the correct

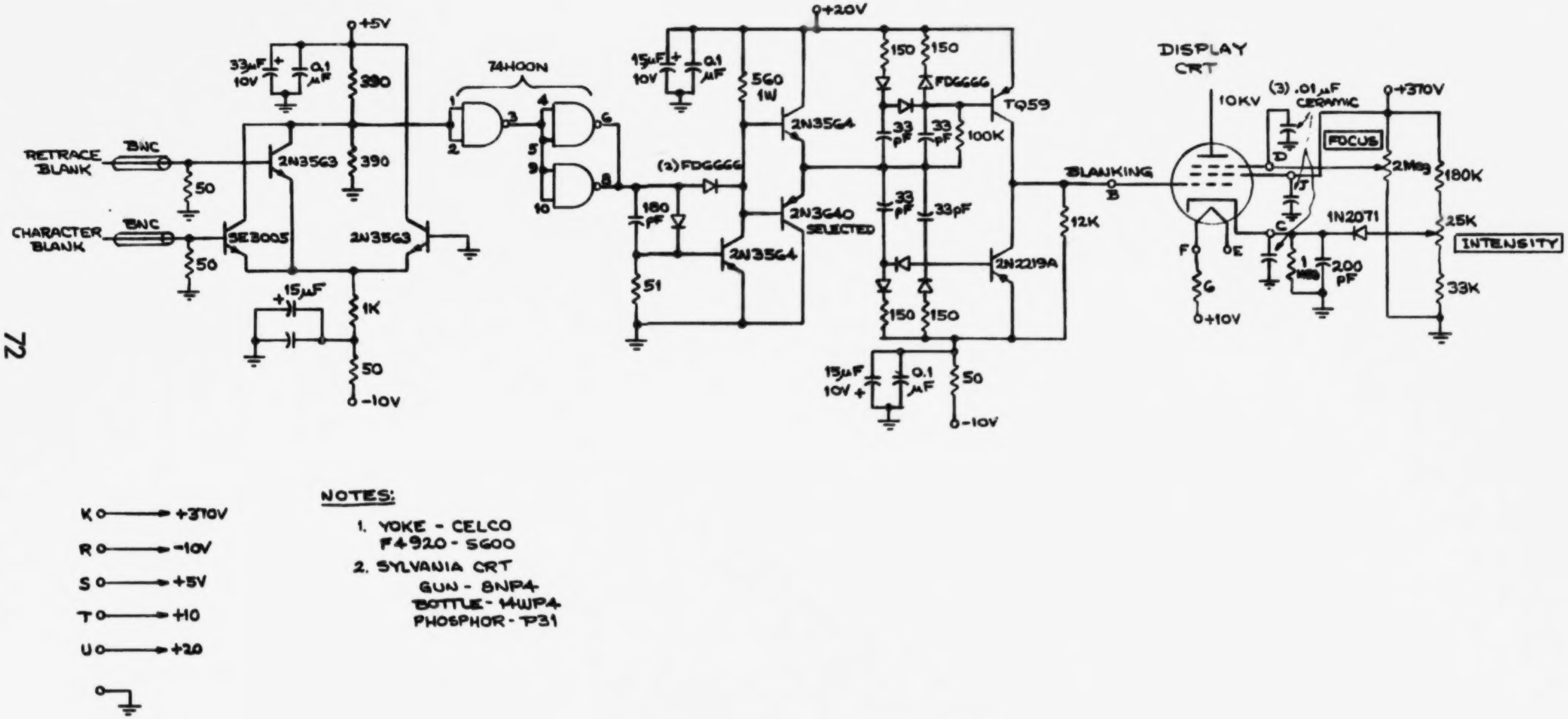

RETRACE BLANK
CHARACTER BLANK
BNC
BNC
50
50
2N3563
SE3005
2N3563
33µF 10V
0.1 µF
+5V
390
390
15µF
1K
50
-10V
74H00N
1
2
3
4
5
6
8
9
10
(2) FD6666
180 pF
51
15µF 10V
0.1 µF
560 1W
2N3564
2N3640 SELECTED
2N3564
+20V
150
150
FD6666
33 pF
33 pF
100K
33 pF
33pF
150
150
TQ59
2N2219A
12K
15µF 10V
0.1 µF
50
-10V
BLANKING
B
DISPLAY CRT
10KV
(3) .01µF CERAMIC
D
J
C
F
E
G
+10V
1 Meg
200 pF
1N2071
+370V
FOCUS
2Meg
180K
25K
INTENSITY
33K
K → +370V
R → -10V
S → +5V
T → +10
U → +20
NOTES:
1. YOKE - CELCO F4920 - S600
2. SYLVANIA CRT
GUN - 8NP4
BOTTLE - 14WP4
PHOSPHOR - P31

Fig. 4.16 Blanking Amplifier

number of column intervals. Each of these display regions is then segmented by the "vertical generator" (Section 4.2.4) into dot-time intervals.

4.2.1 HORIZONTAL TIMING

It is now necessary to fix the number of columns generated per complete character. When displaying a full screen of characters, it is necessary to leave some space between characters to prevent them from running together. One solution is to add a staircase signal to the ramp before it goes into the horizontal deflection amplifier. The staircase would advance to a new step after each character and move the scan pattern right for the next character. Unfortunately, this scheme requires a settling time of approximately one half of the sine wave period and is therefore impossible with a 10μs deflection system.

Another more tractable method of spacing is to keep the constant sinusoidal scan across the line and leave blank scans between characters. It is desirable to use an even number of scans to plot a character and its space, since then an integral number of sine wave periods would be used, and all characters would begin on the same phase of the sine wave. Because there are an odd number of scans (9) consumed in plotting the 13 by 9 matrix, there must also be an odd number of blank scans. One scan gives insufficient space, but three blank scans yield the proper spacing. It is undesirable to go to five or more blank scans, because the percent of time used for plotting is decreased, and hence frequencies in the interpolator must go up. With three blank scans, a total of 12 scans, or six sine wave periods, are necessary per character.

It is therefore necessary to generate six cycles of a sine wave per character. Unfortunately, this frequency is not present in the internal timing of the console. There are periodic pulses available at $10 \cdot f_o$, which are used to shift ten bit ASCII codes from the drum to the console. (f_o is the frequency of plotting characters; $f_o = 0.118$MHz.) These pulses are of no help, since 6 does not divide ten evenly. The only solution is to synthesize a $6 \cdot f_o$ square wave from the f_o pulses present in the console.

4.2.2 PHASE-LOCKED LOOP

Figure 4.17 shows the details of the phase-locked loop shown in the lower left hand corner of Figure 4.1. It generates a frequency 6 times f_o from a reference signal at point A of f_o. (See Figure 4.17.) A voltage-controlled oscillator runs at about $6 \cdot f_o$. The output is divided by 6 to produce a signal at point B of about f_o. This signal is compared to the reference frequency at point A and if it is not the same, an error signal is sent to the integrator and the frequency of the voltage-controlled oscillator is changed appropriately.

It is the phase of the reference signal at A that is compared with the phase of the generated signal at point B. The comparison is done by a 4-state up-down counter which is incremented at every pulse from A, and decremented by every pulse from B. The counter stays in its highest state, called state 3, if it is incremented while in that state--it does not go to its lowest state as a modulo 4 counter would do. It also stays in the lowest state, called state 0, if decremented in state 0. If the counter were started in state 1, the number in the counter at any subsequent time would measure the difference in the number of pulses at point A from the number at point B, provided this difference never exceeds -1 or +2.

The output of the counter is an analog -1 unit if the counter is in state 0 or 1, and is an analog +1 unit if the counter is in state 2 or 3.

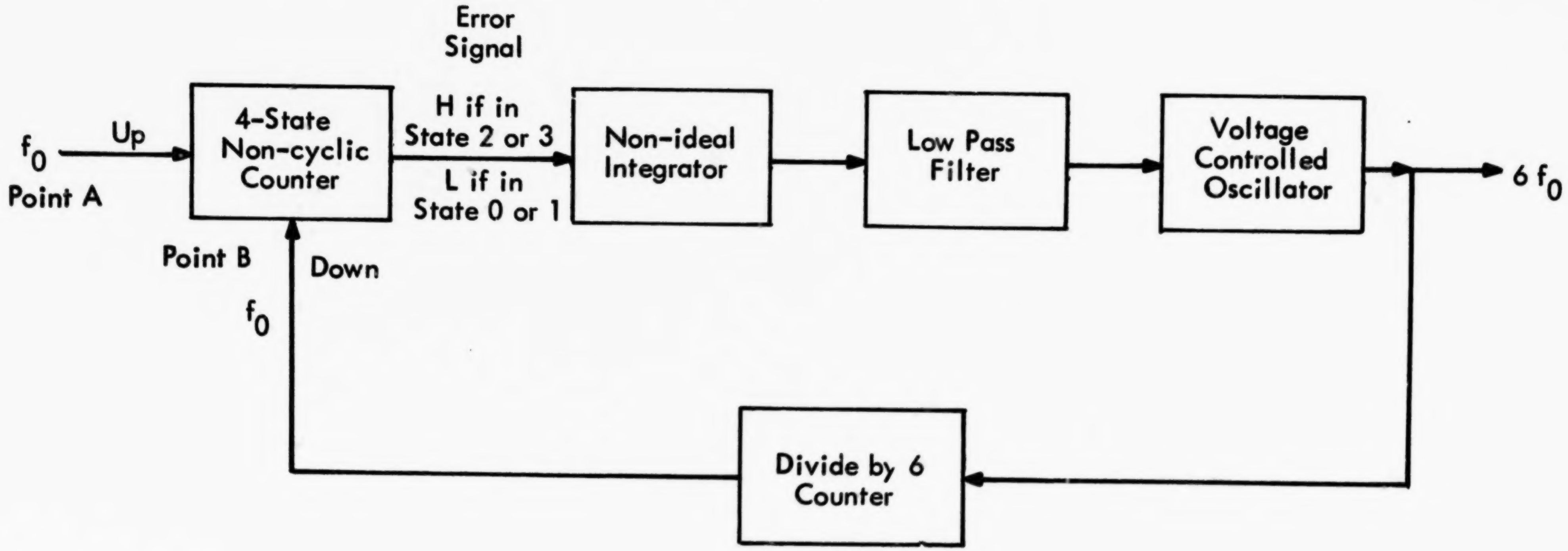

Error
Signal
f_0
Up
Point A
4-State Non-cyclic Counter
H if in State 2 or 3
L if in State 0 or 1
Non-ideal Integrator
Low Pass Filter
Voltage Controlled Oscillator
$6 f_0$
Point B
Down
f_0
Divide by 6 Counter

Fig. 4.17 Phase-Locked Loop

Therefore the integrator's output has a negative slope if the counter contains a 0 or 1 and has a positive slope if the counter contains a 2 or 3. The output of the integrator is fed to a low pass filter, which rolls off with two poles ($Q = 5$) at about $f_o/10$. The filter reduces the pulse to pulse frequency modulation of the oscillator.

There are two distinct modes of operation of the loop, each characterized by the counter being in a different set of states. First is the capturing mode. If the synthesized frequency at point B is much larger (smaller) than the reference frequency f_o at A, then the counter will spend all of its time in states 0 and 1 (states 2 and 3). The integrator would therefore decrease (increase) its value, and the oscillator will also decrease (increase) its frequency. Finally, the frequency at B will actually be littler smaller (larger) than that of A. Two pulses at A (B) will occur between pulses at B (A), and the loop enters state 2 (state 1) from state 1 (state 2).

The second mode is called the locked mode, and is characterized by the counter being in state 1 or 2. To show that the loop does indeed lock when in states 1 or 2, it is necessary to get the open loop frequency response of the system. Figure 4.18 shows the frequency response of the various parts of the system. Two of the boxes require some special explanation. The first is the comparator. When operating exclusively in states 1 and 2, pulses come to the comparator alternately from points A and then from B. One pulse from A takes the comparator from state 1 to state 2, and then a pulse from B takes it from state 2 to state 1. The time average of the output of the counter, taken over one of these cycles, is zero if the two signals are exactly 180 degrees out of phase. The time average, taken in

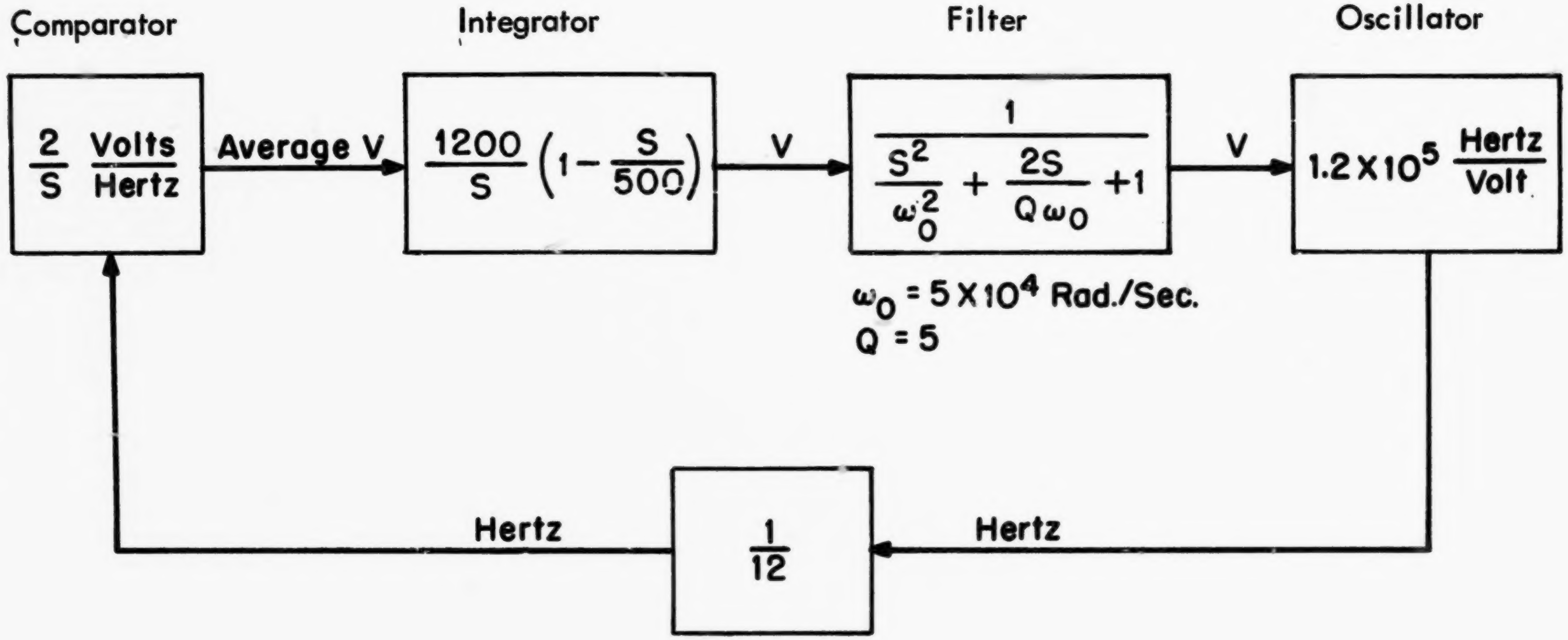

Comparator
$\frac{2}{S}\ \frac{\text{Volts}}{\text{Hertz}}$
Average V
Integrator
$\frac{1200}{S}\left(1-\frac{S}{500}\right)$
V
Filter
$\frac{1}{\frac{S^2}{\omega_0^2}+\frac{2S}{Q\omega_0}+1}$
$\omega_0 = 5 \times 10^4$ Rad./Sec.
$Q = 5$
V
Oscillator
$1.2 \times 10^5\ \frac{\text{Hertz}}{\text{Volt}}$
Hertz
$\frac{1}{12}$
Hertz

Fig. 4.18 Frequency Diagram of Phase-Locked Loop

the same way, is proportional to the difference of the relative phases from 180 degrees. In terms of frequency of the square wave input, the output is proportional to the integral of the difference of frequencies. Therefore the comparator integrates its input frequency to obtain its output, and is denoted as an integrator.

The other integrator in the system also contains a zero at about 100 Hz. The purpose of the zero is to reduce the phase shift when the loop gain is unity, thus making the loop stable. Figure 4.19 shows the open loop phase and magnitude of the phase locked loop. About 70 degrees of phase margin are present at unity gain.

The loop was designed to have unity gain at about 1kHz. This low crossover frequency is tolerable since reference frequency cannot change very fast. The reference signal is derived from the drum which takes seconds to change its frequency because of its large moment of inertia. The loop also has the advantage of smoothing out any jitter in the drum signals caused by incorrect recording or domain granularities.

The hardware realization of the phase locked loop, along with some circuits to be discussed, is shown in Figure 4.20.

4.2.3 VERTICAL DEFLECTION CIRCUITS

The output of the phase locked loop is low-pass filtered into a sine wave, amplified, and sent to a tuned transformer. (See Figure 4.21.) The output of the transformer is added to the vertical deflection amplifier by placing the output in series with the deflection amplifier. The sum is sent directly to the vertical coils of the display CRT. This vertical signal along with the horizontal ramp, produce the sinusoidal scan pattern.

Although the sinusoidal scan does bypass the settling time of the

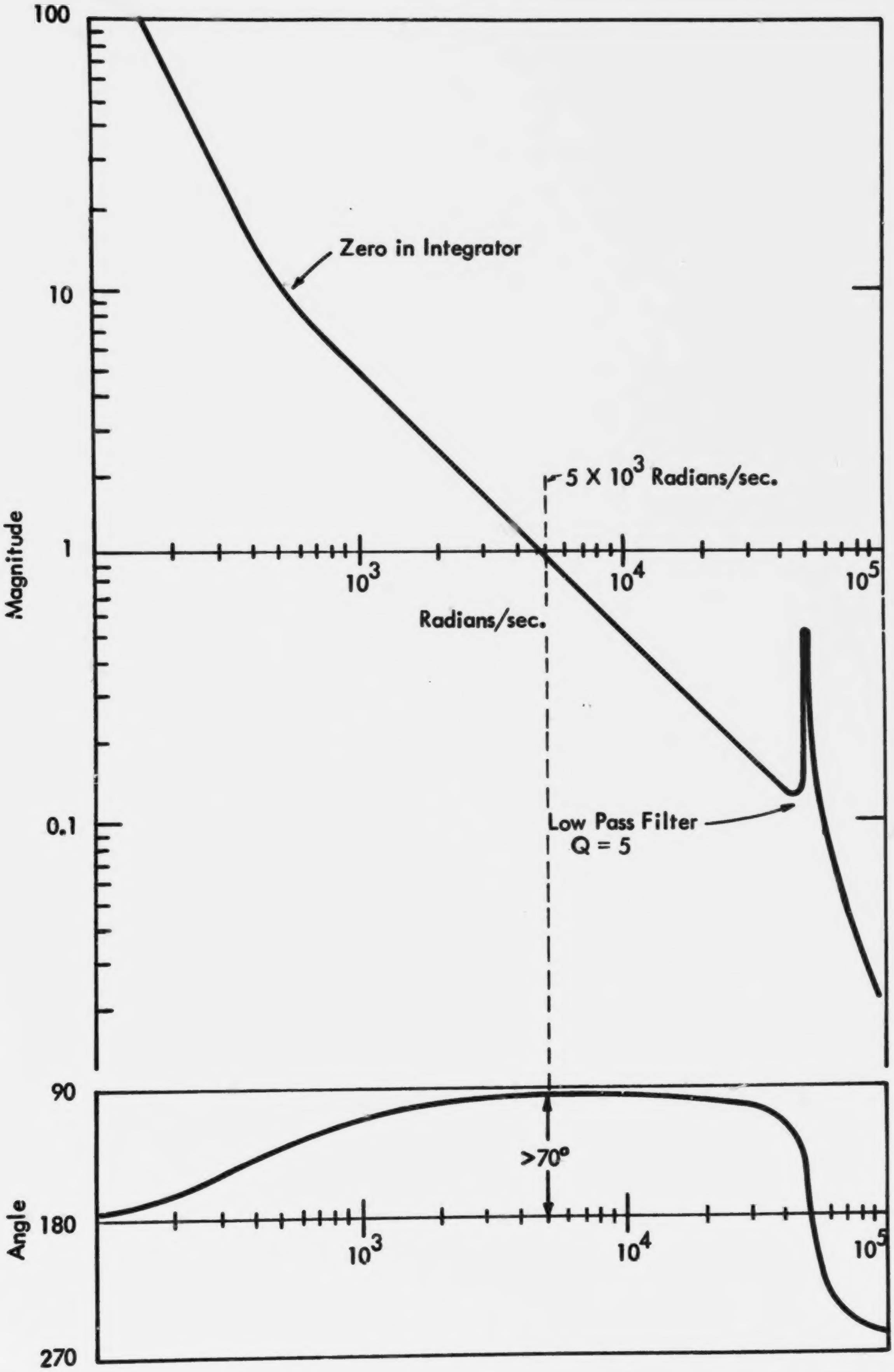
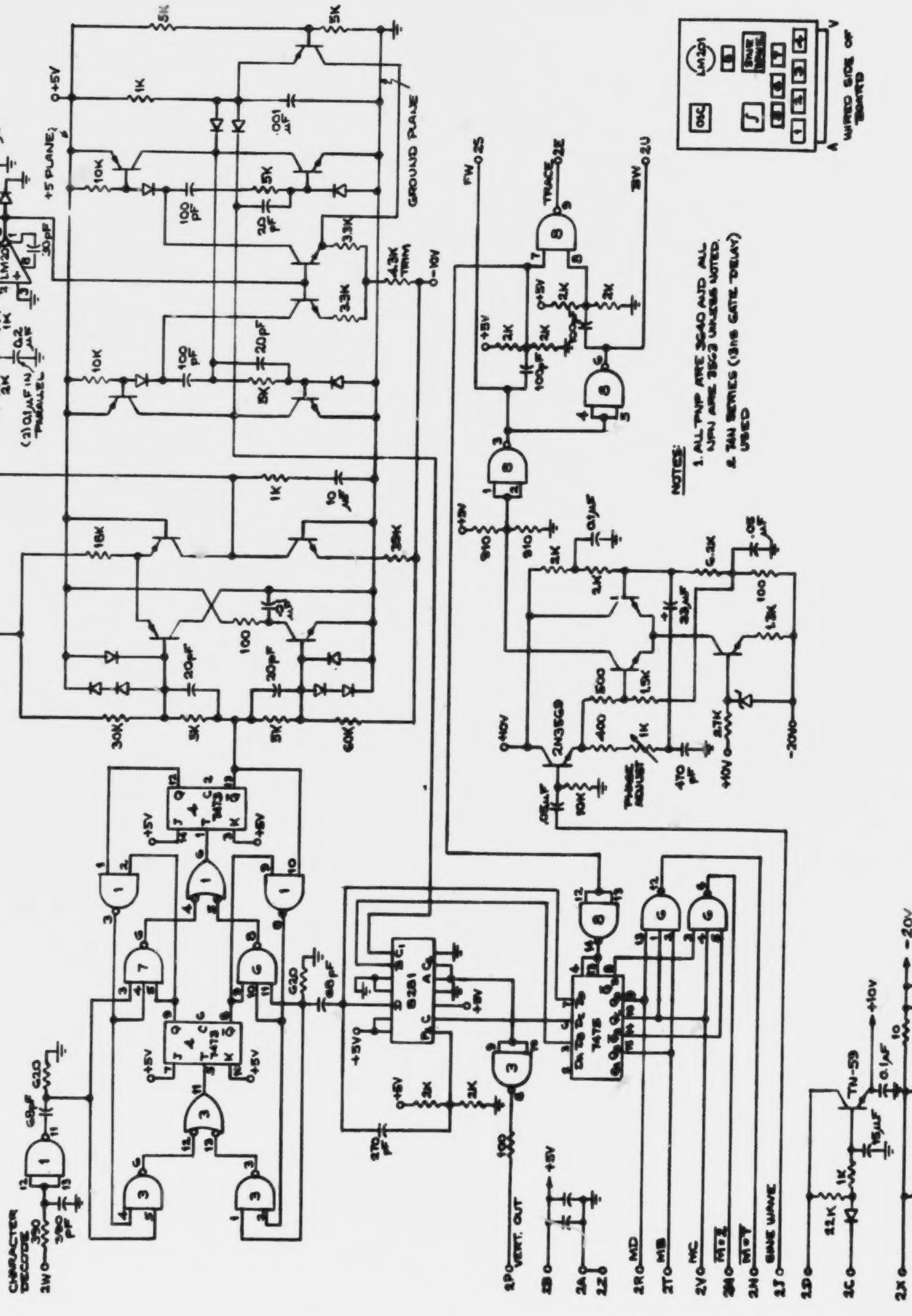


Fig. 4.19 Magnitude and Phase of Open Loop


CHARACTER DECODE
2W
330
330 pF
2B
2A
2Z
1P
VERT. OUT
2R
MD
2T
MB
2V
MC
2M
2N
SINE WAVE
2J
1D
1C
2X
2Y
7473
74123
8281
7475
2N3563
2N3563
LM201
+5 PLANE
GROUND PLANE
+10V
-10V
+5V
-20V
30K
5K
60K
16K
39K
10K
1K
2K
.001 µF
100 pF
20 pF
33K
4.3K TRIM
TN-59
TO-59
22K
470 pF
DELAY
PHASE ADJUST
TRACE
FW
2S
2E
2U
BW
WIRED SIDE OF BOARD
OSC
LM201
SINE WAVE

NOTES:
1. ALL PNP ARE 3G40 AND ALL NPN ARE 3G3 UNLESS NOTED.
2. 74H SERIES (13ns GATE DELAY) USED

Fig. 4.20 Horizontal Generator

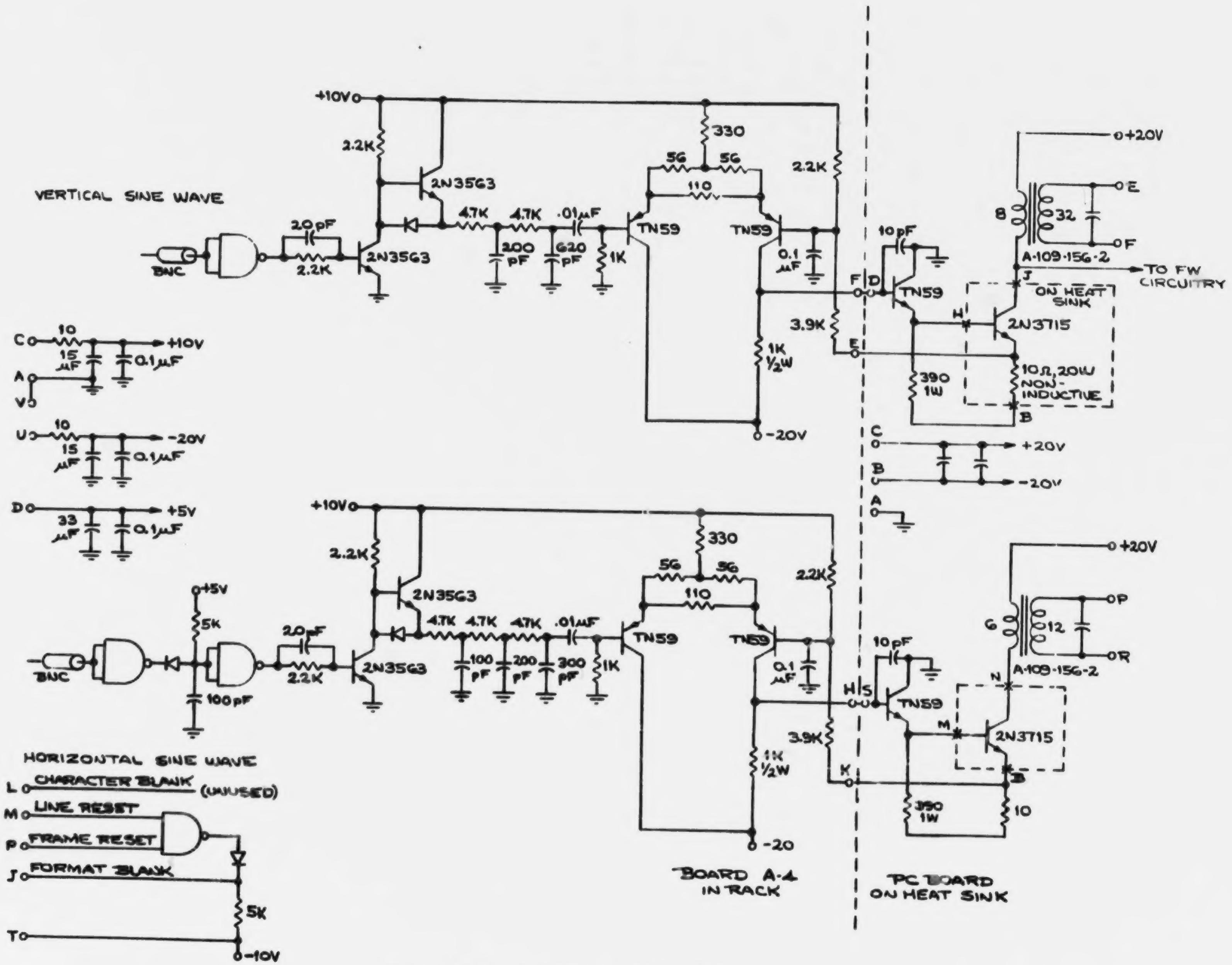

VERTICAL SINE WAVE
BNC
20pF
2.2K
2N3563
+10V
4.7K
200 pF
620 pF
.01µF
1K
TN59
330
56
110
0.1 µF
3.9K
1K ½W
-20V
F
D
E
10pF
H
J
2N3715
ON HEAT SINK
390 1W
10Ω, 20W NON-INDUCTIVE
B
8
32
A-109-156-2
+20V
TO FW CIRCUITRY
C
A
+20V
-20V
C
A
V
U
D
10
15 µF
0.1µF
+10V
-20V
33 µF
+5V
HORIZONTAL SINE WAVE
+5V
5K
100pF
100 pF
300 pF
S
K
M
N
6
12
P
R
10
-20
BOARD A-4 IN RACK
PC BOARD ON HEAT SINK
L
CHARACTER BLANK
(UNUSED)
M
LINE RESET
P
FRAME RESET
J
FORMAT BLANK
T
5K
-10V

Fig. 4.21 Tuned Amplifiers

vertical system, it does not allow accurate positioning of the beam. This is an inherent problem with the sinusoidal scan which becomes very important at this point--the phase delay of the vertical sinusoid amplifier is not constant. The main problem is that the sine wave must pass through a high Q transformer before it is added to the vertical raster signal. Small frequency shifts in a high Q system about its resonant frequency produce large frequency shifts. Both the input frequency and the resonant frequency vary. The scan frequency varies because the timing for the whole system is ultimately derived from a clock track recorded on the drum. Jitter on the clock tracks or hunting of the drum causes the frequency of the sine wave to change. The resonant frequency of the amplifier changes because the inductance of the yoke is in parallel with the tuned transformer. Since the yoke is non-linear, its inductance, and hence the resonant frequency of the transformer change depending on the position of the character on the screen. Changes of the scan frequencies about the resonant frequency of the toroidial transformer cause a large phase shift. In addition the frequency response of the amplifier can not be altogether stable, because of temperature drifts and stray capacitance. These arguments imply the position of the beam can not faithfully follow the sine wave used to drive it.

Although it is not possible to predict the position of the beam knowing only the input to the amplifier, it is possible to sense the beam position by measuring the yoke voltage. To derive this relation, recall that 1) the angle of deflection in a CRT is proportional to the B-field in the yoke; 2) the B-field is proportional to the yoke current; and 3) the change in current in the yoke is proportional to the voltage across the yoke. These statements together imply that the voltage across the yoke is proportional to the velocity of the beam.

The voltage across the yoke is therefore fed to a differential amplifier and clipped about its zero crossings. The output of the clipper (called FW which stands for a forward direction of scan) becomes ONE when the beam is at its maximum height, and stays ONE while the beam is scanning downward. FW becomes ZERO when the beam is at its lowest point, and stays ZERO while the beam is scanning upward.

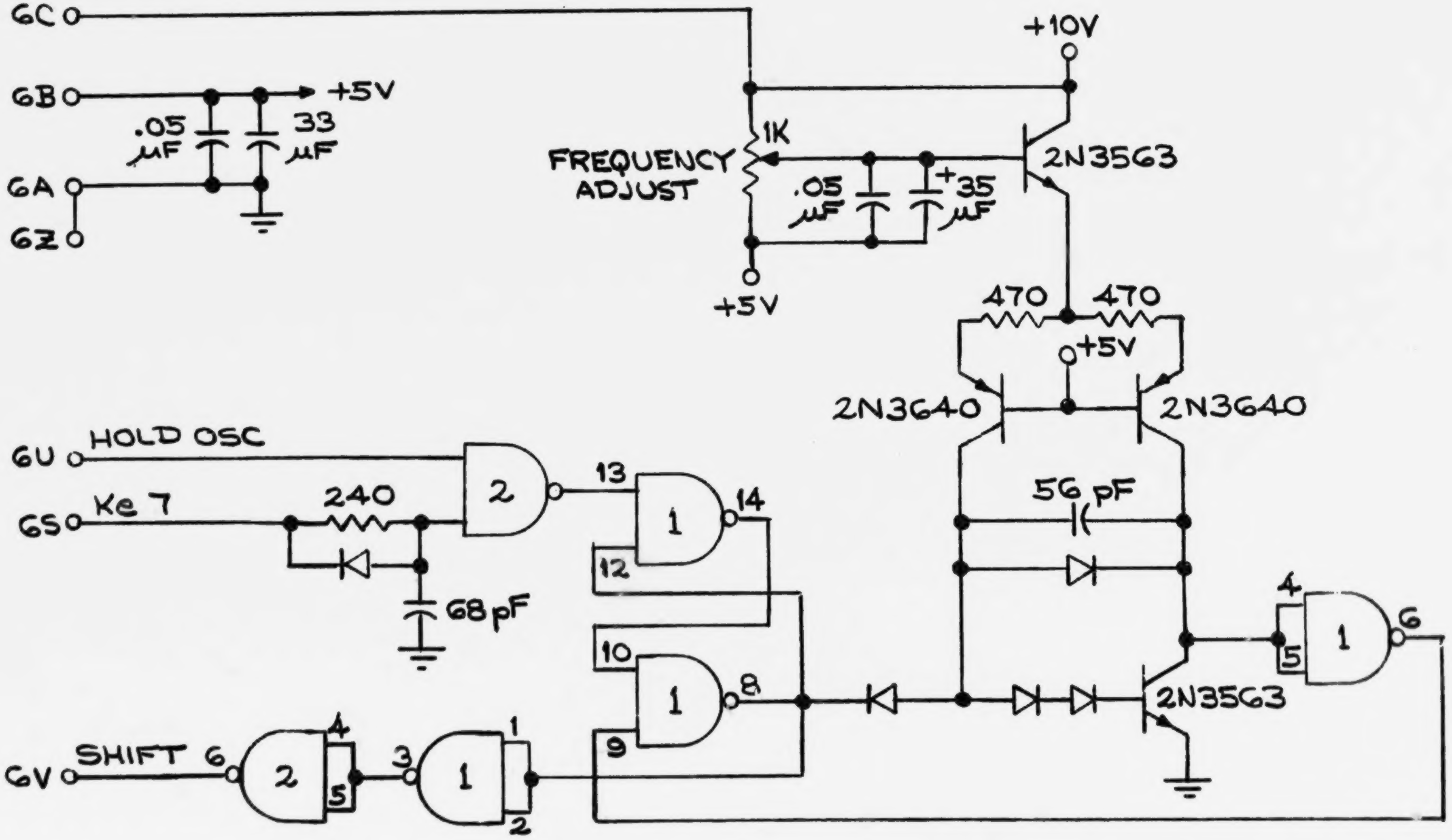
All the internal timing signals of the interpolator-formattor are derived from the FW signal, because it is the only accurate measure of the beam's position. The phase-locked loop output, which drives the beam, cannot be used because it jitters with respect to FW.

FW itself is used to tell the direction of the scan, and hence whether the scan is a primary or secondary column. The falling edge of FW is used to gate the state of the divide-by-six counter in the phase-locked loop into a three-bit latch. This register contains the number of the current column being generated. A latch had to be used to force the column count to change synchronously to the change of FW.

A signal called TRACE is formed from FW. TRACE has a pulse whenever FW changes from ONE to ZERO or from ZERO to ONE. TRACE is used to order-flip the shift-registers, since this must happen before each scan. (See Section 4.1.4.) If TRACE is delayed by the appropriate amount, it occurs at the beginning of the region used to display the matrix. The delayed TRACE pulse starts an oscillator which starts generating SHIFT pulses.

4.2.4 VERTICAL TIMING

The SHIFT pulses are generated by a 20MHz oscillator shown in Figure 4.22. SHIFT pulses do three things. Primarily, they are used to shift the


6C
6B
+5V
.05 µF
33 µF
6A
6Z
FREQUENCY ADJUST
1K
.05 µF
+35 µF
+10V
2N3563
+5V
470
470
+5V
2N3640
2N3640
56 pF
6U
HOLD OSC
6S
Ke 7
240
68 pF
2
13
1
14
12
10
1
8
9
6V
SHIFT
6
2
4
5
3
1
1
2
2N3563
4
5
1
6

NOTE: ALL DIODES 1N125

Fig. 4.22 SHIFT Oscillator

shift registers which generate the window matrix (Figure 4.13). Secondly, they are used to reclock the ouput of the combinational logic which does the interpolation (Figure 4.15). On the leading edge of SHIFT, the primary dot is outputted to the Z-axis of the CRT. On the falling edge the secondary dot is outputted. The third task of the SHIFT pulses is to drive an m-state counter which tells what bit in the current column is being generated (on Figure 4.13). When this counter is zero or m-1, one of two levels is sent to the window generator to signal that certain bits of the window matrix are invalid (see page 62). The counter also turns off the oscillator after m shift pulses have been generated. The above process causes m SHIFT pulses to occur after each delayed TRACE pulse. If the frequency of the oscillator is right, the m'th pulse will occur exactly at the end of the display region.

4.3 READ-ONLY MEMORY

The previous section described the hardware necessary to transform a skeletal matrix into a display matrix, and to plot that matrix on the CRT. It was assumed throughout those sections that the interpolator was somehow supplied with the correct skeletal matrix. This section describes the process which supplies the interpolator with the appropriate skeletal matrix.

4.3.1 MEMORY OF THE SCANNING GENERATORS

To better understand the work of the digital character generator, consider the operation of the scanning generators of Chapter I. These generators use the character code to determine where on the mask the beam is to be positioned. The method used to find a character is both simple and elegant. There is a simple 1-1 map from the ASCII code onto the array of characters on the mask. The following map will illustrate the point. If an ASCII code has the binary expansion:

$$b_8 \cdot 2^7 + b_7 \cdot 2^6 + b_6 \cdot 2^5 + b_5 \cdot 2^4 + b_4 \cdot 2^3 + b_3 \cdot 2^2 + b_2 \cdot 2^1 + b_1 \cdot 2^0$$

then the code is mapped into row:

$$b_4 \cdot 2^3 + b_3 \cdot 2^2 + b_2 \cdot 2^1 + b_1 \cdot 2^0$$

and column:

$$b_8 \cdot 2^3 + b_7 \cdot 2^2 + b_6 \cdot 2^1 + b_5 \cdot 2^0$$

The elegant feature of the system is that when the mask is made, the characters are arranged so that each character is under the position into which the code for that character maps. Characters are stored on the mask in a manner which makes their retrieval very easy.

Consider the mask and the scanning beam of the scanning generator to be a physically constructed read-only memory. The code then specifies the address in the memory where the image of the code is to be found. In an analogous manner, the digital character generator has the skeleton matrix of each of its characters stored in a digital read-only memory. The code from the drum is used to determine where in this memory the matrix of the character represented by that code is found.

The problem is to find a simple map which will allow the skeletal matrices for all the characters desired to be stored in the memory available. First, it is necessary to define the problem more explicitly. The following paragraphs describe the method of storing the matrices, and the structure of the codes used to address them.

4.3.2 STORING SKELETAL MATRICES

The organization of the skeletal matrices in the ROM is a compromise between two extremes. One extreme is to have 35 bits per word in the memory, and thus one whole matrix would fit into one word of the memory. This requires handling 35 bits of information in parallel somewhere in the machine. Duplicating data paths 35 times, even if for only one register, is an unpleasant task. A more serial approach seems better. The other extreme in organization is totally serial. Consider storing only one bit per word. Thus 35 words are necessary to store one skeletal matrix. Unfortunately, this method requires 35 cycles of the memory to retrieve each character, and forces the memory to operate at unrealistic speed.

It was decided to store each matrix in five words, with each word containing seven bits. Each word is one column of the matrix, and thus the memory scheme is compatible with the format required in Section 4.1. This organization also eliminates the problems associated with the totally serial and the totally parallel structures of the previous paragraph.

4.3.3 MODIFIED ASCII CODE

As with the scanning generators, all information which is to be displayed on the CRT screen is stored in ASCII-coded form on the magnetic drum. Characters are displayed on the CRT screen in the same order that they are recorded on the drum track. As one character is read serially from the drum, it is sent to the console, bit by bit. Once at the console, it is converted to parallel and used to drive the character generator.

The standard ASCII codes received at the console have 10 bits. There are seven bits of coded information, and three bits of header. The header is the same for all characters, so that there are 2^7 or 128 possible codes. The 128 codes are devided into two groups: control codes and visible codes. Visible codes include all letters, numbers, and punctuation marks. Control codes include carriage return, line feed, tab, status, etc., and are characterized by the fact that bit 6 and bit 7 of their code are both zero. Therefore there are 32 control codes, and 96 visible codes in the ASCII character set.

One design objective for the display system is to have a 192-character set. For this reason, a character set flip-flop is added to the console to double the number of visible codes. Depending on the state of this flip-flop, characters are selected from one of two possible character sets. Two control codes in the standard ASCII set are reserved to set and reset the character-set flip-flop. Other control codes operate independently of the character set, since functions like line feed and carriage return are needed in both sets.

The state of this flip-flop is sent to the character generator, along with the seven bit ASCII code. Together they form an eight-bit modified ASCII code. This code has the property that bit 6 and/or bit 7 is always ONE. The character generator ignores codes in which bit 6 and bit 7 are both ZERO: in that case it displays a blank.

4.3.4 ADDRESS MAP

It is necessary to store 192 5-word skeletal matrices in the ROM so that the blocks are easily retrievable. Most memory is produced in units which contain 2^n words, where n is an integer. The problem now becomes that of packing these 960 words into 2^{10} or 1024 words of memory so that access is easy.

The easy access map is motivated by the following factorization of the numbers involved. There are $15 \cdot 2^6$ words to be stored in $16 \cdot 2^6$ locations. Therefore store 15 words, or three skeletal matrices, in each block of 16 locations. Since bits 6 and 7 of the ASCII code together can assume only the three values 01, 10, and 11, (they cannot both be zero), they are a likely candidate to tell which of the three skeletal matrices in the block of 16 words is desired. Then bits 1 through 5 and bit 8 (the character-set flip-flop) tell which of the 2^6 possible blocks of 16 locations is being considered.

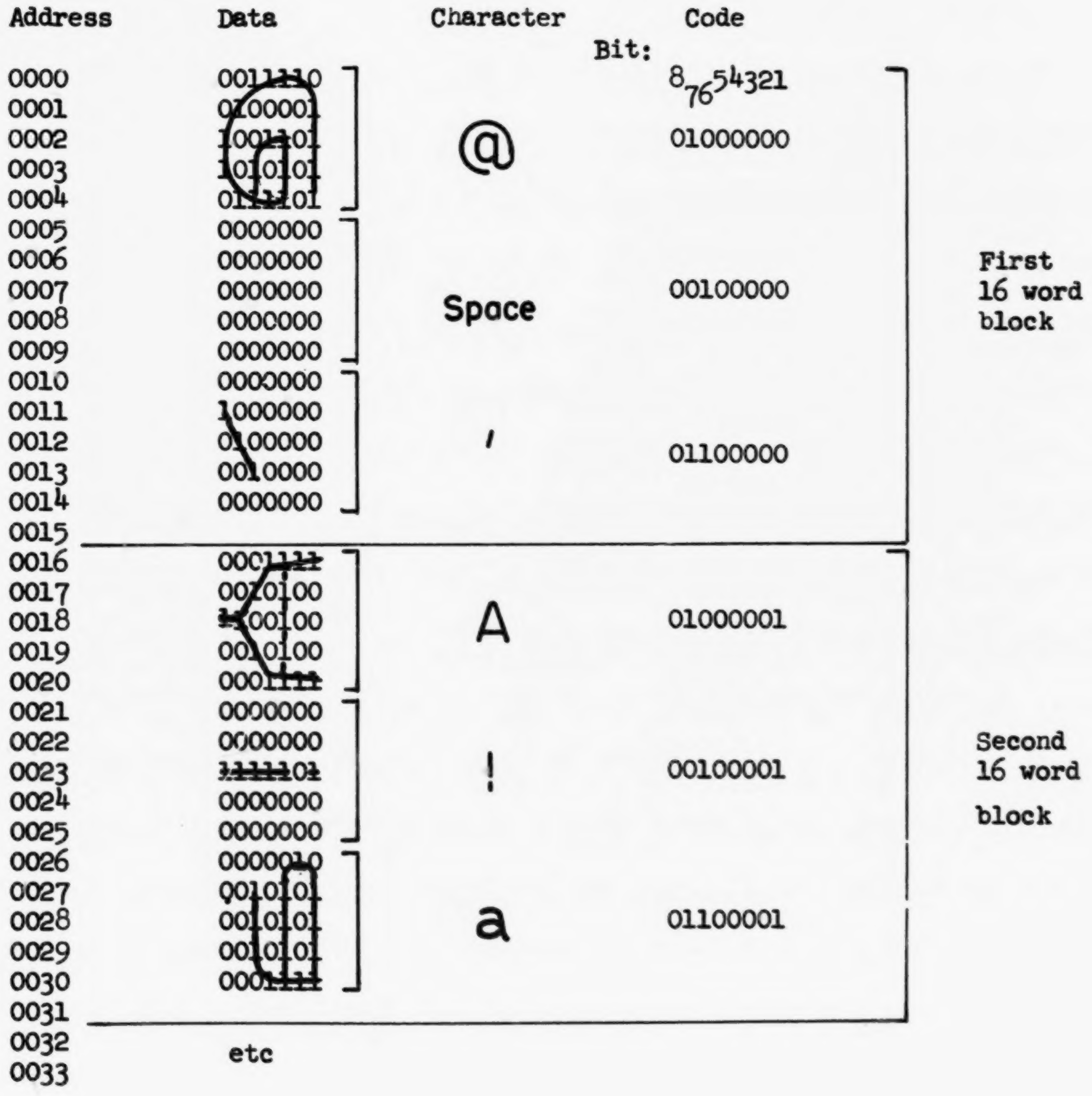
The columns of the skeletal matrices are stored in blocks of five consecutive words. If bit 6 is ZERO and bit 7 is ONE, locations 0 through 4 in the appropriate 16 word block are addressed. If bit 6 is ONE and bit 7 is ZERO, locations 5 through 9 in the 16 word block are addressed. Finally, if both bits 6 and 7 are ONE, locations 10 through 14 are addressed. Location 15 of the block is not used presently, although a use for it will be described in Section 4.3.5.

To implement the above in hardware, a sixteen-state counter is used. The counter is preset to either 0, 5, or 10, depending on the values of bit 6 and bit 7. The counter is then indexed to obtain the next column in the skeletal matrix. Figure 4.23 shows the contents of the first 32 locations in the read-only memory.

This mapping has one pleasing feature: it is simple. It requires only three NAND gates and one four-bit counter. For implementation, it does not require that the whole memory address register of the ROM be a counter. All columns for a character are in the same block of 16 words. Only four bits of the memory address must count.

4.3.5 KEY WORD

The sixteenth word in each block is used in the following manner. The bits of this word can be divided amongst the three skeletal matrices stored in the block. Two tag bits per matrix can be stored in this last word. The first tag tells what type of interpolation is to be applied to the matrix. The second tag could be used to tell the vertical position at which the matrix should be plotted on the CRT. If the character should extend below the line (e.g. g,j,p,q,y), the 7 by 5 matrix is lowered to make the character extend below the line. The shift could be achieved by delaying one edge of FW (Section 4.2.3) for regular characters, and the other



Address	Data	Character	Code	
			Bit:	
0000	0011110		87654321	
0001	0100001			
0002	1001101	@	01000000	
0003	1010101			
0004	0111101			
0005	0000000			
0006	0000000			
0007	0000000	Space	00100000	First 16 word block
0008	0000000			
0009	0000000			
0010	0000000			
0011	1000000			
0012	0100000	'	01100000	
0013	0010000			
0014	0000000			
0015				
0016	0001111			
0017	0010100			
0018	1000100	A	01000001	
0019	0010100			
0020	0001111			
0021	0000000			
0022	0000000			
0023	1111101	!	00100001	Second 16 word block
0024	0000000			
0025	0000000			
0026	0000010			
0027	0010101			
0028	0010101	a	01100001	
0029	0010101			
0030	0001111			
0031				
0032	etc			
0033				

Figure 4.23 Contents of memory

edge of FW for low characters. Thus the display matrix could be shifted downward for low characters.

This second tag was not used in the prototype character generator. Since the memory which was purchased had eight bits per word, and since the interpolator was initially constructed for an eight-bit column, an 8 by 5 skeletal matrix was used. The bottom row of the matrix is always zero for regular characters, and the top row is always zero for low characters. Thus the non-zero portion of each character is actually contained in a 7 by 5 matrix.

It is necessary to retrieve the key word for a character before that character is plotted. As explained in Section 4.2.1, six cycles of the sinusoidal scan are generated on the CRT to plot one character. Only five of these sinudoids are actually used to plot points: one scan is always blank. A read-only memory cycle must precede each of these five plotting cycles to retrieve the new column which the interpolator needs to begin that scan. During the sixth cycle of the sinusoid, the interpolator needs no new data. The read-only memory fetches the key word for the next code to be displayed at that time.

The interface for the read-only memory is shown in Figure 4.24. It contains the address map, the memory address register/counter, and the timing necessary to make the memory operate in the manner described.

An additional feature was added to the system to facilitate the process of changing a skeletal matrix. One can select any modified ASCII code on a set of eight switches. Whenever the character generator is to generate this selected character, it reads the skeletal matrix from an array of 8 by 5 switches, instead of from the ROM. For the one selected code, any

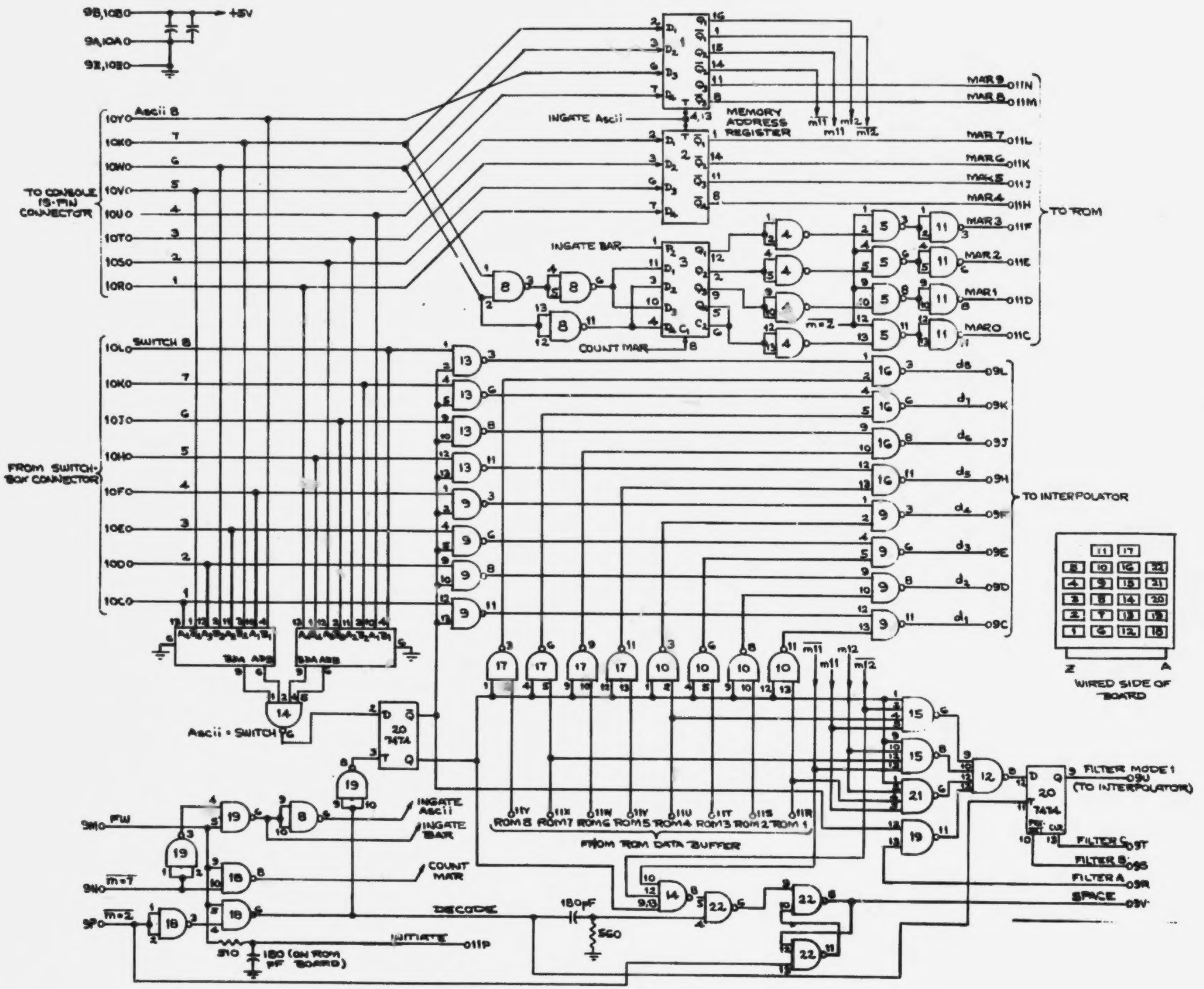

9B,10B
9A,10A
9Z,10Z
+5V
TO CONSOLE 15-PIN CONNECTOR
Ascii 8
FROM SWITCH-BOX CONNECTOR
SWITCH 8
MEMORY ADDRESS REGISTER
INGATE Ascii
INGATE BAR
COUNT MAR
MAR 9
MAR 8
MAR 7
MAR 6
MAR 5
MAR 4
MAR 3
MAR 2
MAR 1
MAR 0
TO ROM
m11
m12
m=2
TO INTERPOLATOR
Ascii = SWITCH
FW
m=7
m=2
INGATE Ascii
INGATE BAR
COUNT MAR
DECODE
INITIATE
180 (ON ROM PF BOARD)
180pF
560
ROM 8 ROM 7 ROM 6 ROM 5 ROM 4 ROM 3 ROM 2 ROM 1
FROM ROM DATA BUFFER
FILTER MODE 1 (TO INTERPOLATOR)
FILTER C
FILTER B
FILTER A
SPACE
7474
WIRED SIDE OF BOARD
Z
A

Fig. 4.24 ROM-Interpolator Interface

skeletal matrix can be fed to the interpolator in place of the one stored in the ROM.

This feature proved to be quite valuable. It allows one to change any one character of the generator in seconds. The new character is then viewed in the context in which it appears on the page being viewed. If the character contained in the switches is superior to the one in the ROM, the ROM can then be changed. Single words can be altered in the Wire-Rope memory with about fifteen minutes work. Alternatively, the corrections can be recorded, and a completely new memory contents can be purchased. The cost of a new memory contents is a small fraction of the cost of the whole ROM system.

CHAPTER V

CONCLUSION

The previous chapters have analyzed the problem of character generation on a bandwidth-limited CRT, and proposed a method of generating high quality characters on such a system. Figure 5.1 shows a picture of the Intrex console, with the character generator indicated. This chapter will discuss the results of the character generator developed.

5.1 COMPARISON OF DOT-MATRIX AND SCANNING GENERATORS

Figures 5.2 through 5.9 are actual photographs of the face of the CRT. Figure 5.2 shows the character set produced by the dot-matrix character generator. Compare this picture with Figure 5.3 and 5.4 which shows the monoscope characters. Figure 5.3 was taken using the interlace technique mentioned in Chapter I. Although the picture cannot show this, the effective refresh rate is 28.5 cycles per second,producing noticeable flicker. It can be eliminated by removing the interlace as shown in Figure 5.4. It should be noted that the refresh rate of both the dot-matrix character in Figure 5.2 and monoscope character in Figure 5.4 is 57 cycles per second. It is clear from the pictures that the dot-matrix characters have much better resolution. In addition, the jitter associated with the scanning generator is not present. This elimination of jitter cannot be shown in the picture, but is evident when viewing the CRT.

Pictures of the flying spot scanner have not been included because it was not properly adjusted when the present set of pictures was taken. It is generally agreed that the monoscope characters were of better quality than those of the flying spot scanner.

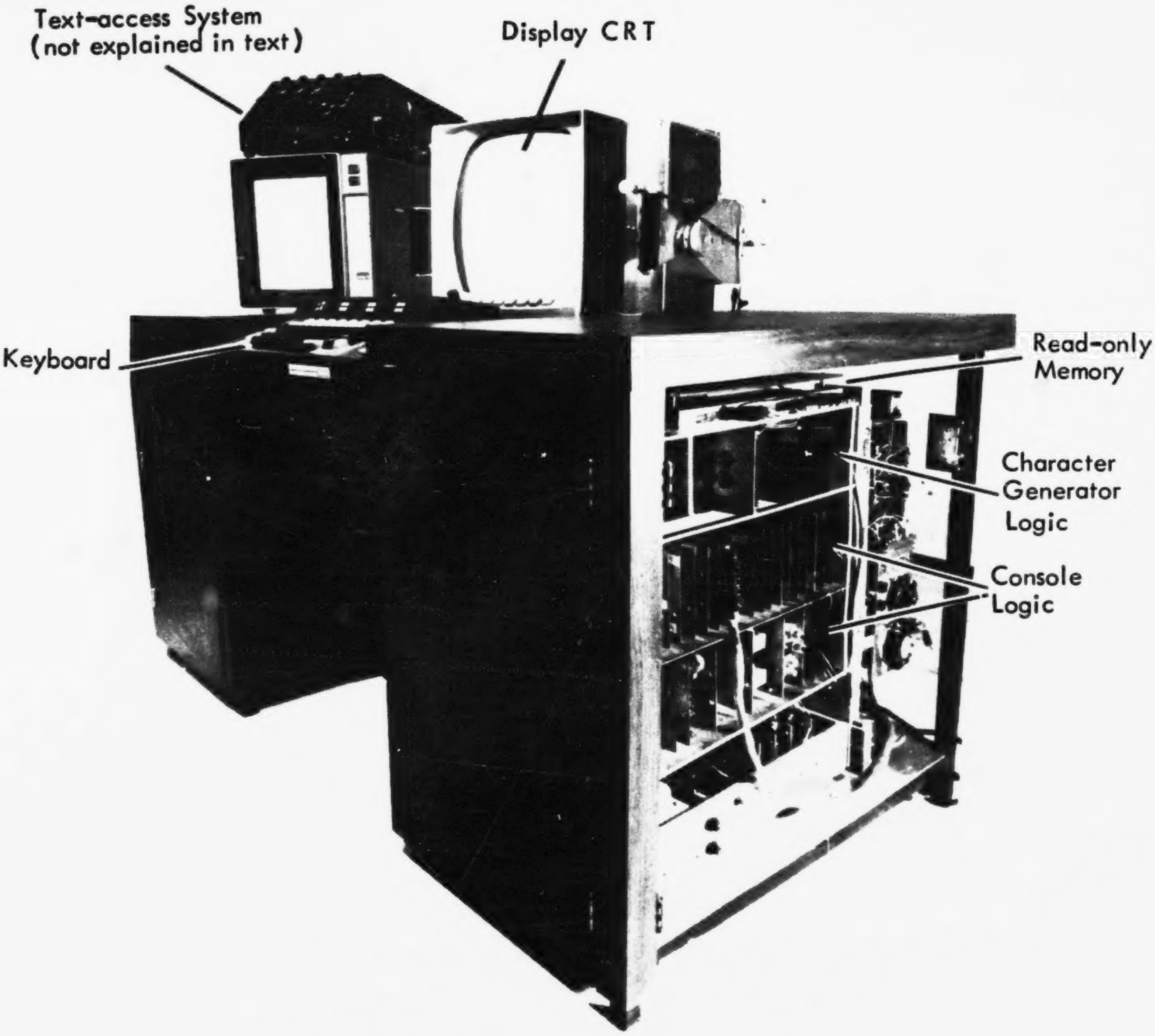

Text-access System
(not explained in text)
Display CRT
Keyboard
Read-only Memory
Character Generator Logic
Console Logic

Fig. 5.1 The Intrex Console

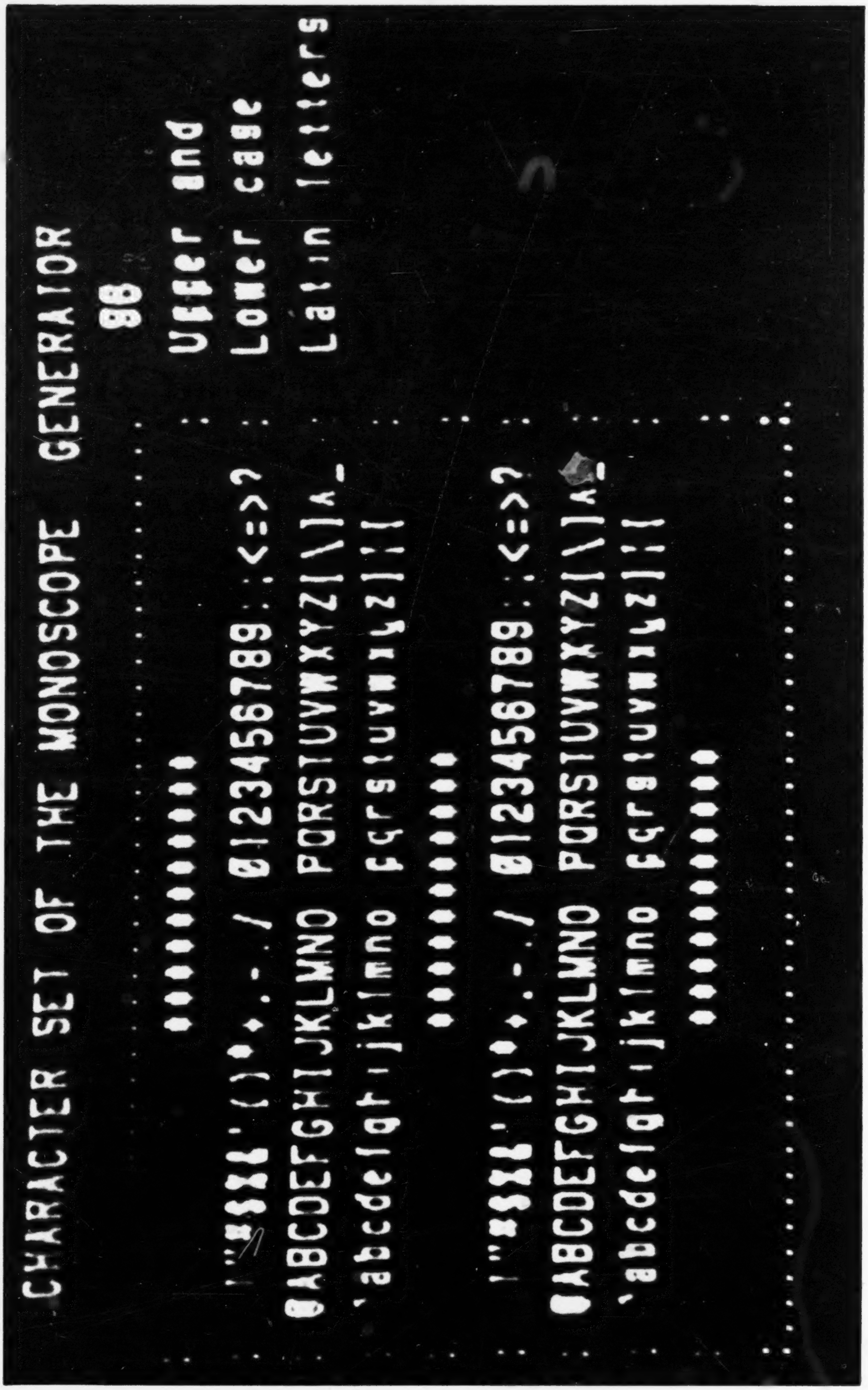

CHARACTER SET OF THE MONOSCOPE GENERATOR
98
Uffer and
Lomer case
Latin letters
0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_

Figure 5.3 Monoscope with Interlace

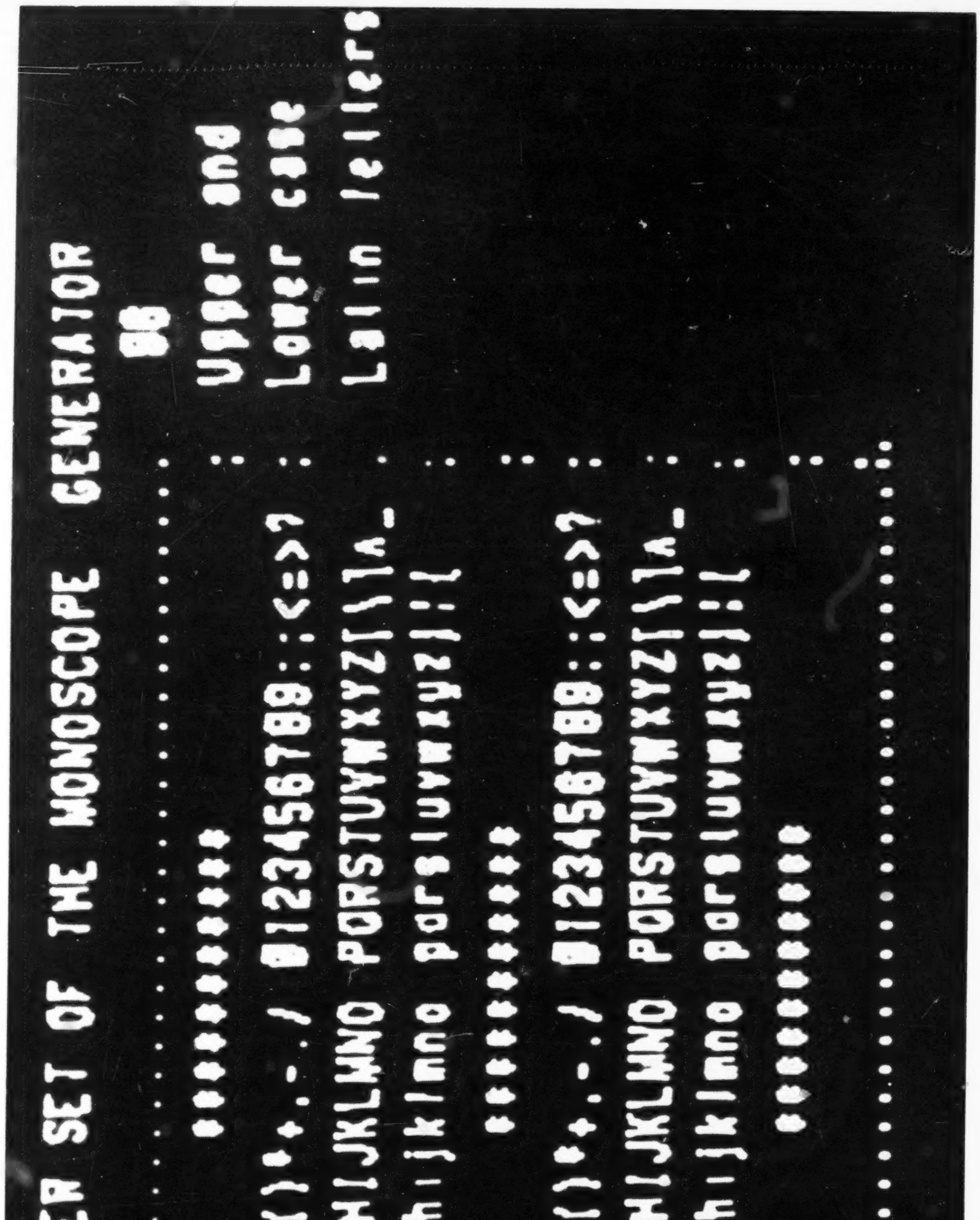

CHARACTER SET OF THE MONOSCOPE GENERATOR
96
Upper and
Lower case
Latin letters
!"#$%&'()*+,-./ 0123456789:;<=>?
@ABCDEFGHIJKLMNO PQRSTUVWXYZ[\]^_
`abcdefghijklmno pqrstuvwxyz{|}~
!"#$%&'()*+,-./ 0123456789:;<=>?
@ABCDEFGHIJKLMNO PQRSTUVWXYZ[\]^_
`abcdefghijklmno pqrstuvwxyz{|}~

Figure 5.4 Monoscope without Interlace

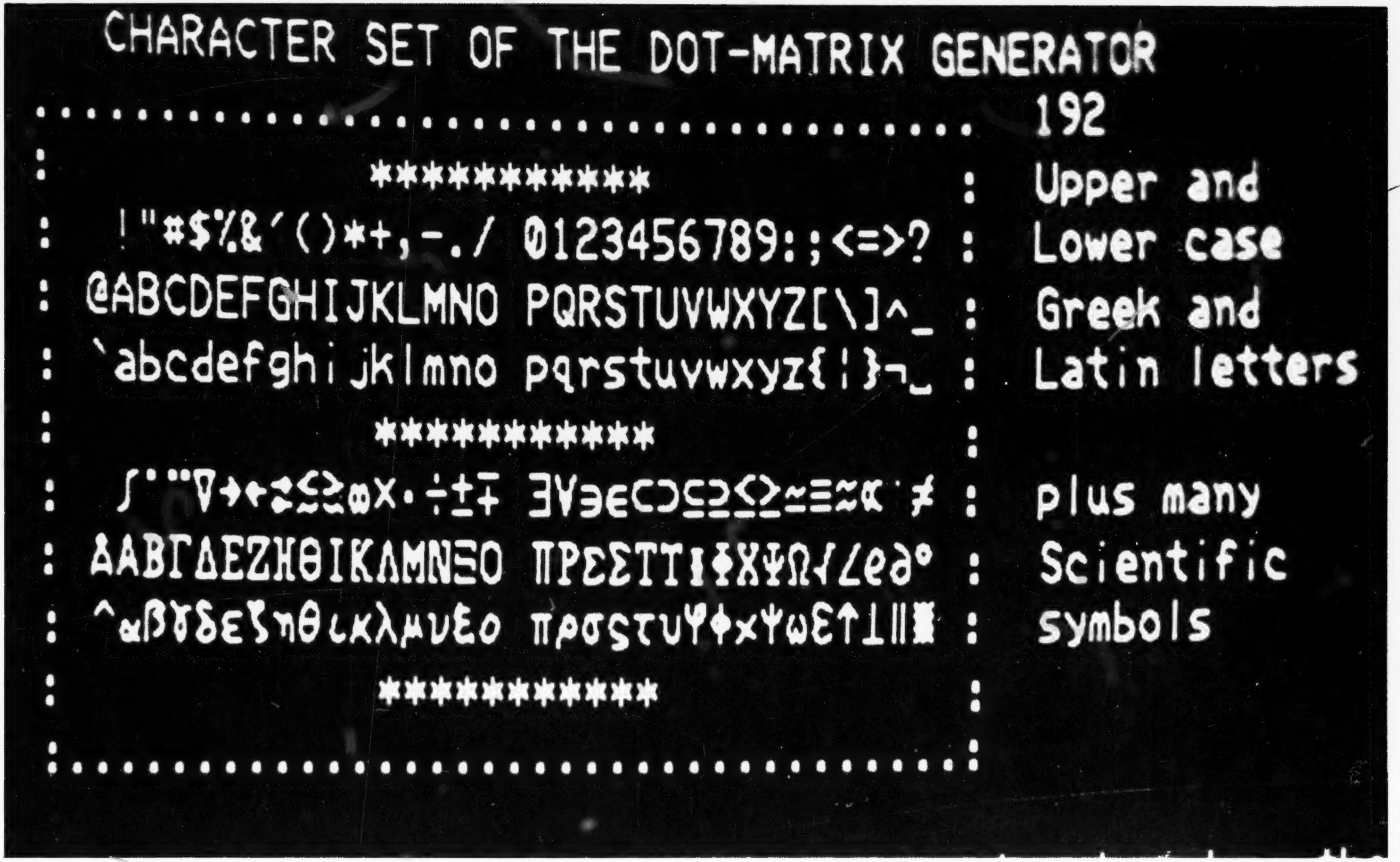
The pictures of the dot-matrix character generator shown in this chapter were displayed on the uncorrected sinusoidal scan, shown in Figure 2.2. The value used for the parameter p was about 0.6. The horizontal double frequency was later implemented, although not in time for pictures. It had very minor effects, and was not included in the final unit.

5.2 EVALUATION OF THE INTERPOLATION ALGORITHMS

Some of the characters of Figure 5.2 were interpolated using simple interpolation and some were interpolated using strong interpolation. A bit signifying the best interpolation type for each character is stored in the ROM (see Section 4.3.5). It is possible to disable the above feature and force all the characters to be interpolated using strong interpolation (Figure 5.5) or simple interpolation (Figure 5.6). Some characters, notably /,7,<,>,@,A,V,W, etc. appear much better in strong interpolation and other characters, notably 8,B,a,b,α,β,Σ,Φ, etc. seem better with simple interpolation. What skeletal matrix is better for each character and which interpolation is best for each matrix is a subjective question and is not explored in this thesis.

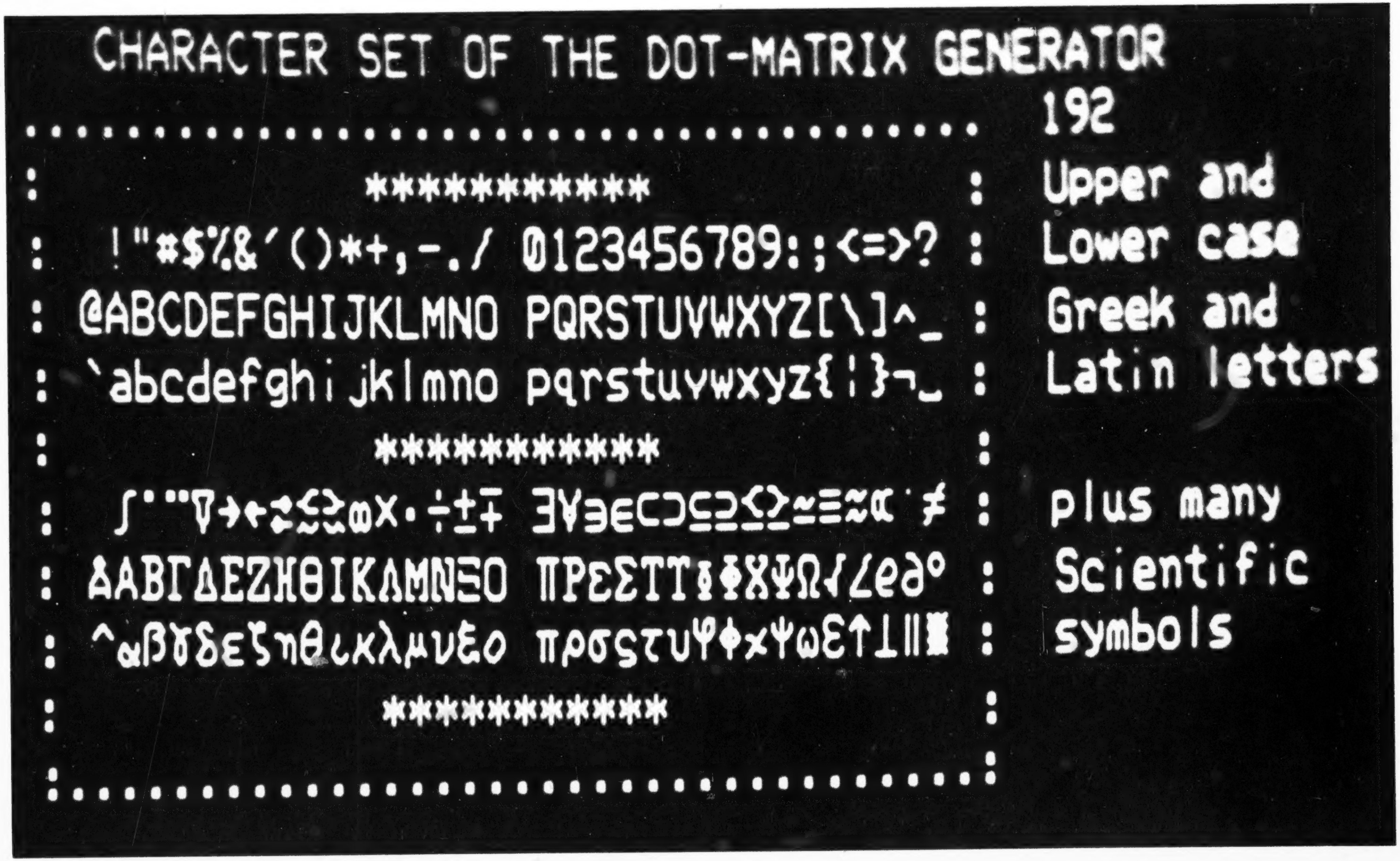
Figure 5.7 shows the dot-matrix characters without any interpolation. The characters are displayed exactly as they are stored in the ROM. This figure should be compared with Figure 5.2 to note the valuable effect of interpolation.

The exact merit of the interpolation algorithm is a subjective question. It is clear that the characters produced are better than 7 by 5 skeletal matrix characters. They also cannot be better than 13 by 9 matrix characters. Many interpolated characters do approach this upper limit of 13



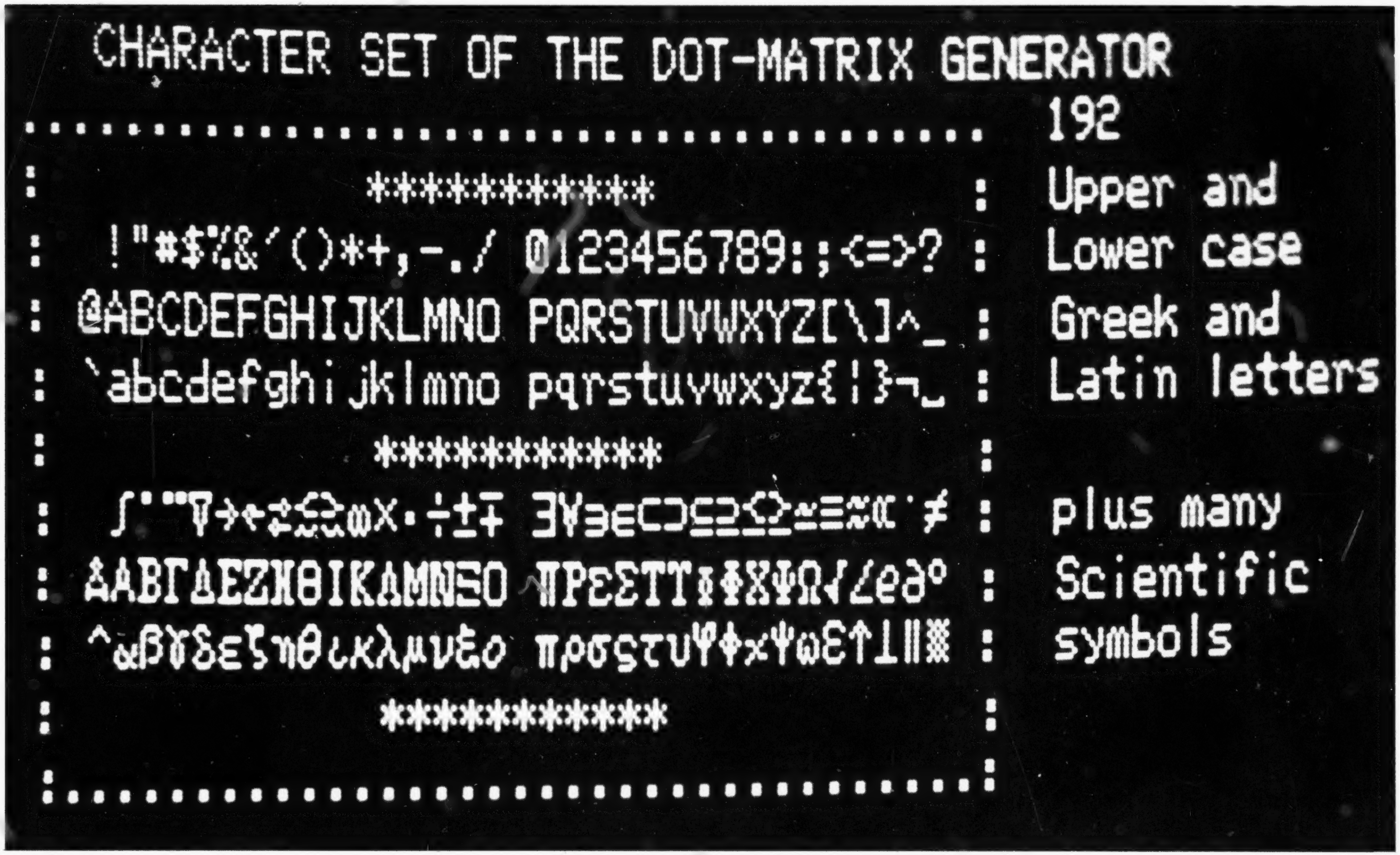
```
CHARACTER SET OF THE DOT-MATRIX GENERATOR
.................................   192
:          ************           :   Upper and
:  !"#$%&'()*+,-./ 0123456789:;<=>? :   Lower case
: @ABCDEFGHIJKLMNO PQRSTUVWXYZ[\]^_ :   Greek and
: `abcdefghijklmno pqrstuvwxyz{¦}¬_ :   Latin letters
:          ************           :
:  ∫˙¨∇→←⇄≦≧∞×·÷±∓ ∃∀∋∈⊂⊃⊆⊇≤≥≃≡≈∝ ≠ :   plus many
: ÅΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟ ΠΡΣΣΤΤΙΦΧΨΩ√∠e∂° :   Scientific
: ^αβγδεζηθικλμνξο πρσςτυΨφχψωϵ↑⊥‖█ :   symbols
:          ************           :
:.................................:
```

Figure 5.5 Dot-Matrix Character Generator; Strong Interpolation



```
   CHARACTER SET OF THE DOT-MATRIX GENERATOR
.............................................  192
:           ***********                     :  Upper and
:  !"#$%&'()*+,-./ 0123456789:;<=>?         :  Lower case
: @ABCDEFGHIJKLMNO PQRSTUVWXYZ[\]^_         :  Greek and
: `abcdefghijklmno pqrstuvwxyz{¦}¬_         :  Latin letters
:           ***********                     :
:  ∫˙¨∇→←⇄≤≥ωx·÷±∓ ∃∀∋∈⊂⊃⊆⊇≪≫≃≡≈∝ ≠         :  plus many
: ÅΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟ ΠΡΕΣΤΤ‡ΦΧΨΩ√∠℮∂°        :  Scientific
: ^αβγδεζηθικλμνξο πρσςτυΨφxψωℇ↑⊥‖▓         :  symbols
:           ***********                     :
:............................................
```

Figure 5.6 Dot-Matrix Character Generator; Simple Interpolation



```
CHARACTER SET OF THE DOT-MATRIX GENERATOR
                                              192
:             ************             :      Upper and
:  !"#$%&'()*+,-./ 0123456789:;<=>? :        Lower case
: @ABCDEFGHIJKLMNO PQRSTUVWXYZ[\]^_ :         Greek and
: `abcdefghijklmno pqrstuvwxyz{|}¬_ :         Latin letters
:             ************             :
:  ∫˙¨∇→←⇄≲≳ω×·÷±∓ ∃∀∋∈⊂⊃⊆⊇≤≥≃≡≈∝˙≠ :        plus many
: ÅΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟ ΠΡΣΣΤΥΦΧΨΩ√∠ə∂° :          Scientific
: ^αβγδεζηθικλμνξο πρσςτυΨφ×ψωƐ↑⊥‖▓ :          symbols
:             ************             :
```

Figure 5.7 Dot-Matrix Character Generator; No Interpolation

by 9 quality characters. The interpolated letter V, for instance, is almost identical to the best letter V which could be displayed on a 13 by 9 matrix. The "smoothness" of characters in general seems to be comparable to the 13 by 9 matrix characters. As noted in Chapter III, the exact positioning of parts of characters is not as precise. The letters b and d, for instance, should have the top of their curved member raised slightly higher in order to conform with the style of the other characters. If the appropriate dots in the skeletal matrix are raised one row, the tops of the curved sections are too high.

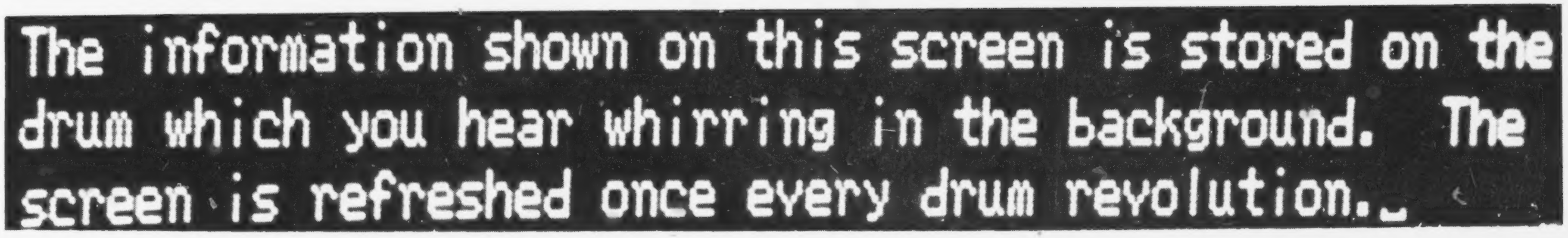
Stated in mathematical terms, the interpolation algorithm transforms a skeletal matrix which has 35 dots into a display matrix which has 117 dots. This display matrix can be said to have about the same quality as a matrix of x times 35 dots, where $1<x<3.3$. One reasonable estimate is that x is approximately 3. That is, one would have to store three times as many dots in the ROM to achieve the same quality characters if the interpolation algorithm is not used.

This value of x allows one to make a direct evaluation of the interpolator. The cost of the character generator, exclusive of the ROM, was about $240.00 in parts (single quantity prices). The cost of the formattor, is about $140.00. Thus the interpolator cost about $100.00 in parts, or say $300.00 fabricated. The interpolator is economically feasible if the ROM economics are such that one could not triple the size of the ROM for less than $100.00 in parts, or $300.00 for a purchased unit.

The interpolation algorithm was found to be economically feasible for MOS and Wire Rope memories, the two types of memories considered. Even with current volume prices for MOS memory, the interpolator costs less than 15% of the memory which it replaces. These prices will not necessarily be valid for the next ten years. Both the cost of the MOS memory, and the cost of the interpolation logic will be reduced. At least for the present, the interpolation algorithm is advantageous.

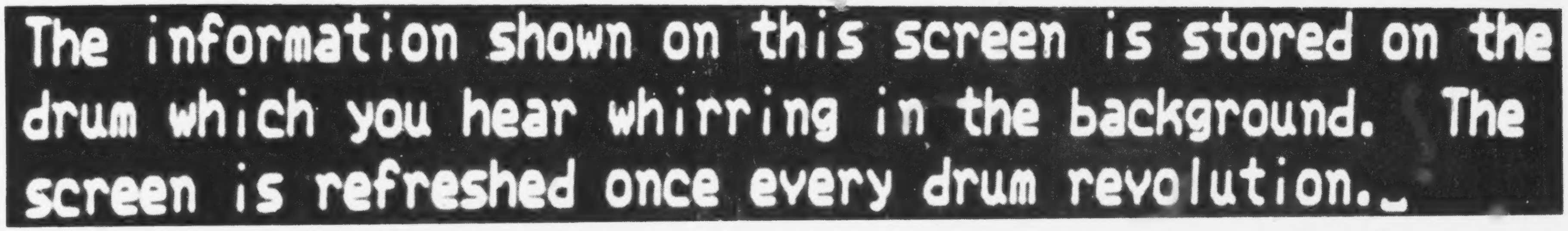
There are other configurations in which the interpolation algorithm may be useful. It may be possible to transmit the small skeletal matrices, instead of the character codes, over the data channel associated with the display. This requires increasing the throughput of the channel by a factor of four, but allows the ROM to be moved to a central location, and hence its cost can be amortized over many displays. In many situations, the ROM might even be eliminated.

In summary, this thesis has demonstrated the feasibility of displaying high quality dot-matrix characters using the low-bandwidth sinusoidal scan. In addition, a technique has been discovered and implemented which reduces the size of the stored character tables associated with dot-matrix displays. The prototype which was constructed incorporates all the ideas mentioned in this thesis (except the vertical correction to the sine wave). The character generator has been used with the Intrex console for about one month, and is fully operational.



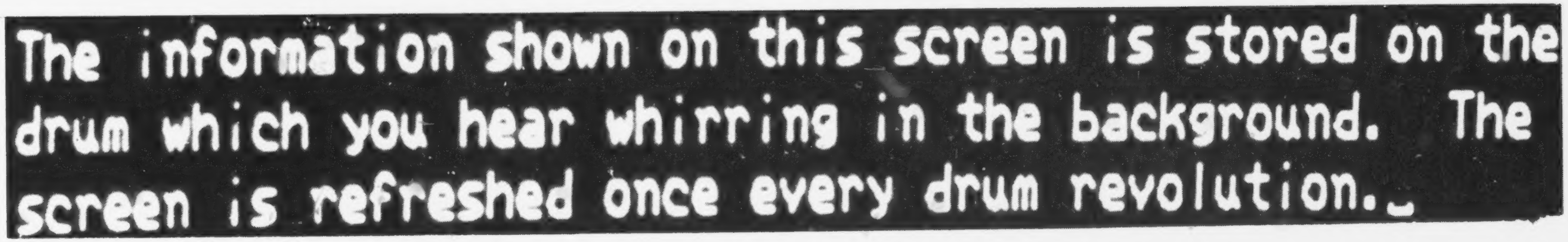
The information shown on this screen is stored on the drum which you hear whirring in the background. The screen is refreshed once every drum revolution._

No Interpolation



The information shown on this screen is stored on the drum which you hear whirring in the background. The screen is refreshed once every drum revolution._

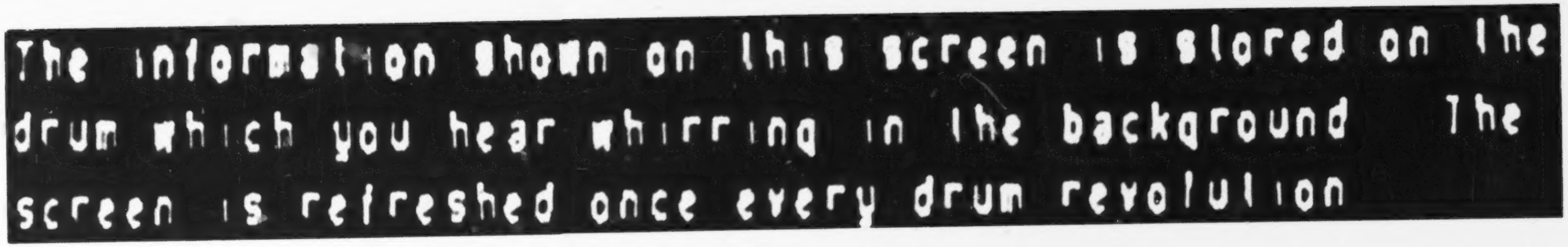
Simple Interpolation Only



The information shown on this screen is stored on the drum which you hear whirring in the background. The screen is refreshed once every drum revolution._

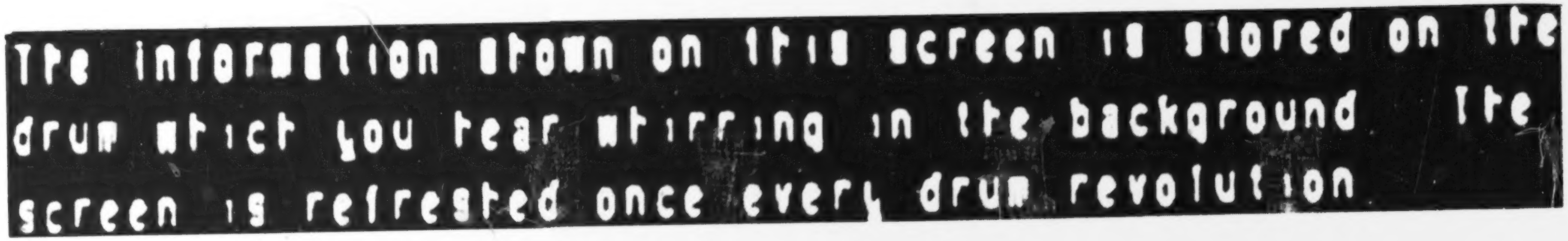
Both Simple and Strong Interpolation

Figure 5.8 Sample Text Using Dot-Matrix



The information shown on this screen is stored on the
drum which you hear whirring in the background The
screen is refreshed once every drum revolution

No Interlace



The information shown on this screen is stored on the
drum which you hear whirring in the background The
screen is refreshed once every drum revolution

Interlace

Figure 5.9 Sample Text Using Monoscope

REFERENCES

1. "Project Intrex Semiannual Activity Reports," 15 March 1967, 15 September 1967, 15 March 1968, and 15 September 1968, M.I.T. Project Intrex, Cambridge, Massachusetts.

2. INTREX, Report of a Planning Conference on Information Transfer Experiment, Eds. Overhage, C. F. J. and Harman, R. J. (The M.I.T. Press, Cambridge, Massachusetts, 1965).

3. Haring, D. R., and Roberge, J. K., Display Techniques for an Experimental Computer-Based Library, October 4, 1968.

4. Haring, D. R., and Roberge, J. K., "The Augmented-Catalog Console for Project Intrex (Part I)," M.I.T. Electronic Systems Laboratory Report ESL-TM-323, October, 1967. (Presented at the IEEE 1967 Lake Arrowhead Workshop on Advanced Computer Peripherals at Lake Arrowhead, California, August 25-27, 1967).

5. McKenzie, P. F., "A Flying Spot Scanner Character Generator," SM Thesis, Massachusetts Institute of Technology, February, 1969.

6. Technical Information Bulletin, Type CK1414 SYMBOLRAY Character Generating Cathode Ray Tube by Raytheon, Components Division, Industrial Components Operation, 465 Centre Street, Quincy, Massachusetts, April 15, 1966.

7. Haring, D. R., "Computer-Driven Display Facilities for an Experimental Computer-Based Library," Proceedings of the 1968 Fall Joint Computer Conference, December 9-11, 1968, San Francisco, California.

8. Haring, D. R., "A Display Console for an Experimental Computer-Based Augmented Library Catalog," Proceedings of the 1968 ACM National Conference and Exposition, August 27-29, 1968, Las Vegas, Nevada, pp. 35-43.

9. Human Engineering Guide to Equipment Design, Eds. Morgan, C. T. Cook III, J. S., Chapanis, A. and Lund, M. W., (McGraw-Hill Book Co., Inc., New York, New York, 1963).

10. Poole, H. H., Fundamentals of Display Systems, (Sparton Books, Washington, D.C., 1966).